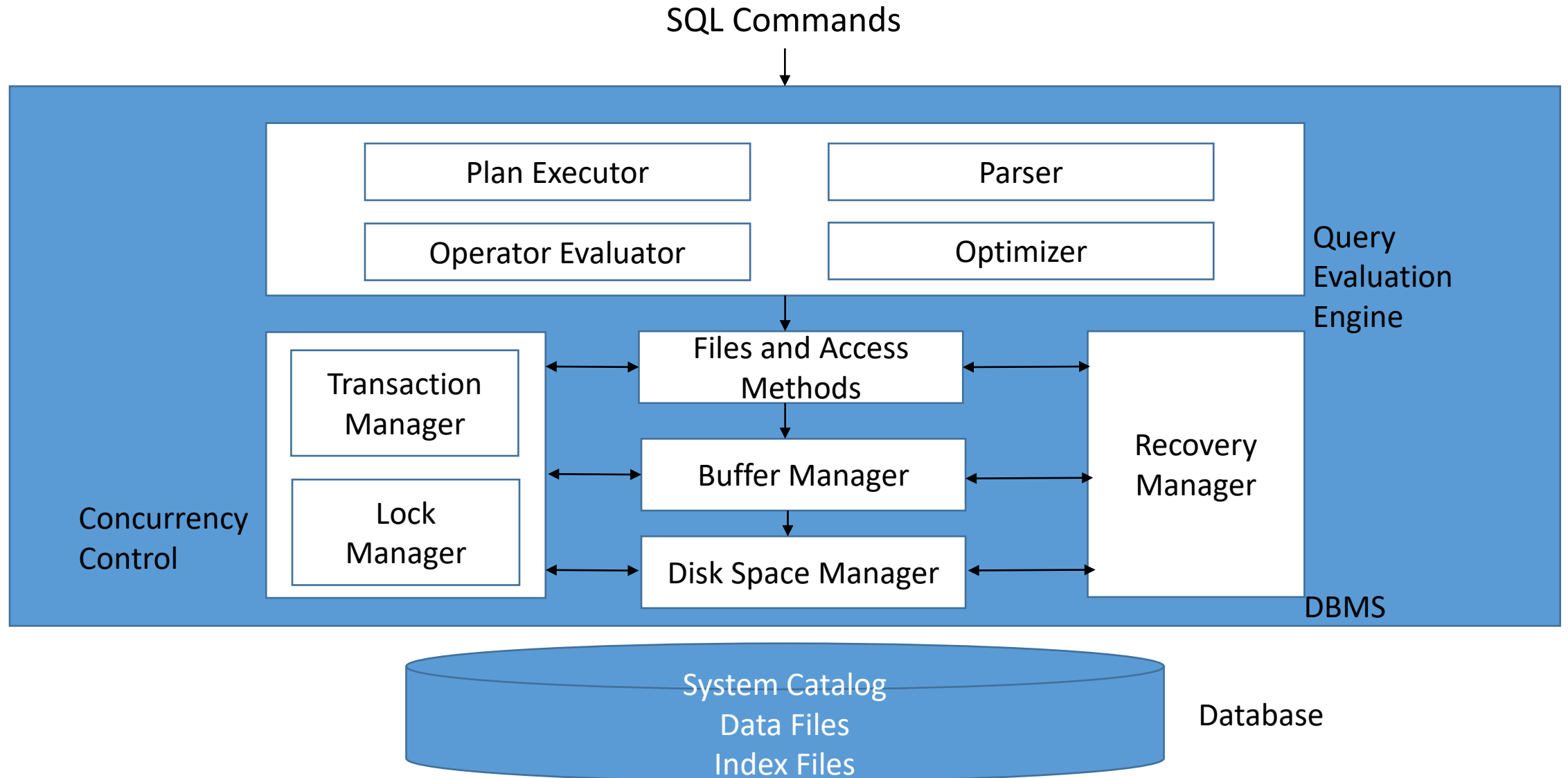


# Databases

## Lecture 8

### The Physical Structure of Databases

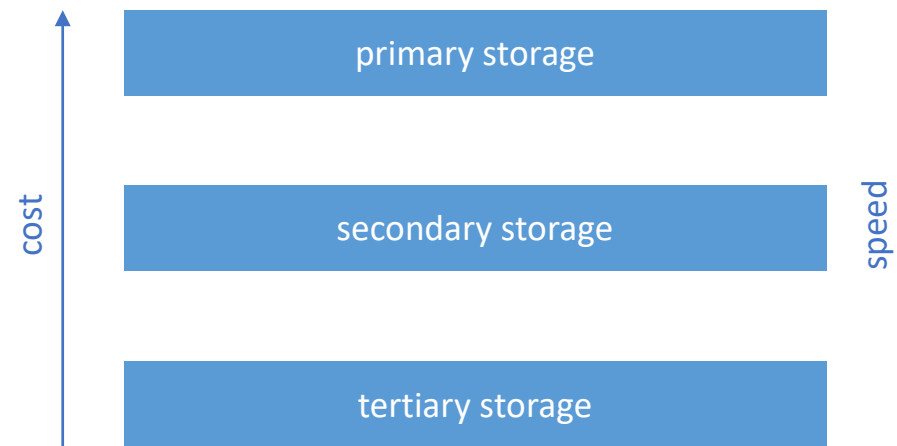
# DBMS Architecture



# The Memory Hierarchy

- primary storage
  - cache, main memory
  - very fast access to data
  - volatile
  - currently used data
- secondary storage
  - e.g., magnetic disks
  - slower storage devices
  - nonvolatile
  - disks - sequential, direct access
  - main database
- tertiary storage
  - e.g., optical disks, tapes
  - slowest storage devices
  - nonvolatile
  - tapes
    - only sequential access
    - good for archives, backups
    - unsuitable for data that is frequently accessed

# The Memory Hierarchy



- disks and tapes - significantly cheaper than main memory
  - large amounts of data that shouldn't be discarded when the system is restarted
- => the need for DBMSs that bring data from disks into main memory for processing

## Secondary Storage – Magnetic Disks

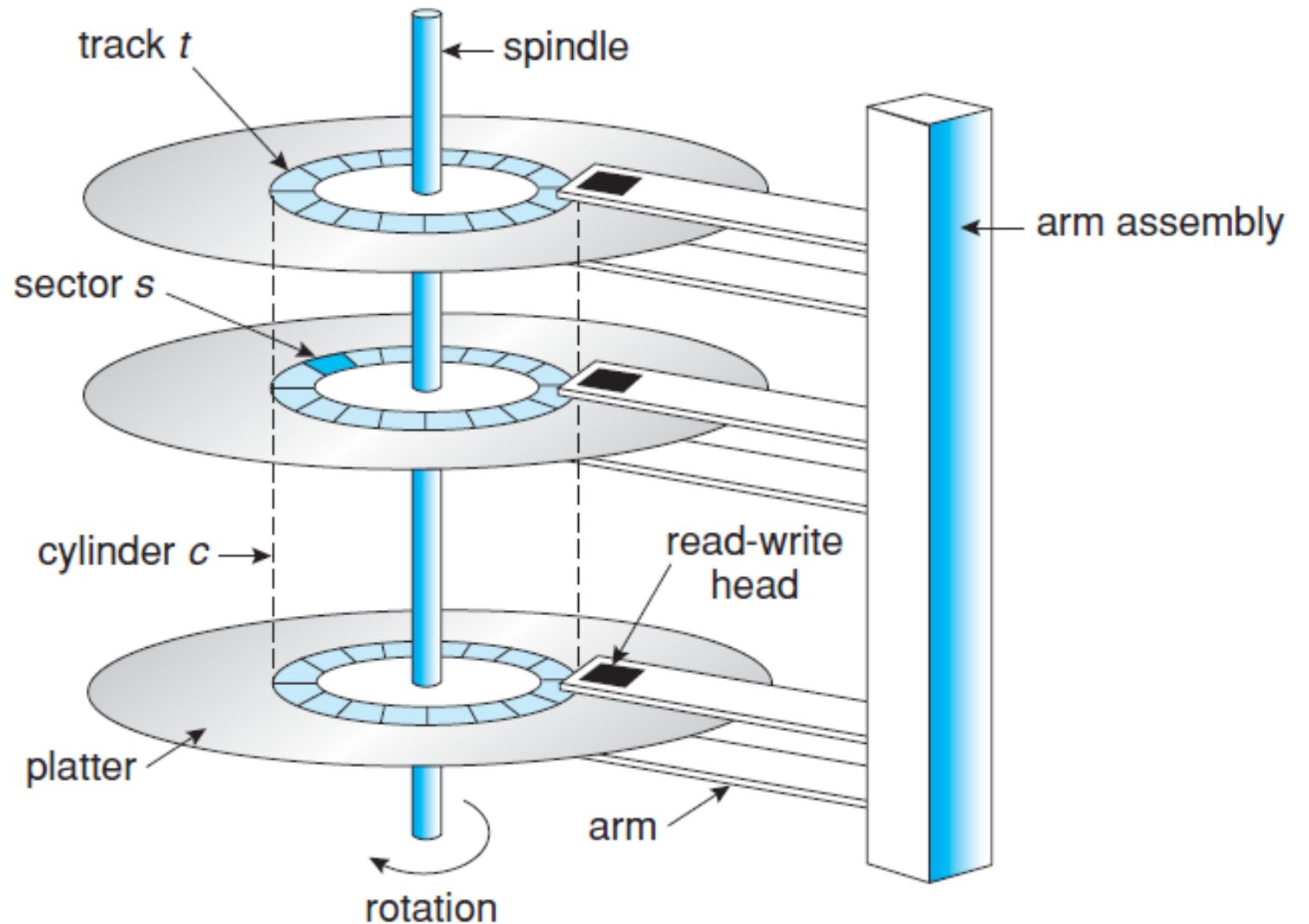
- direct access
- extremely used in database applications
- DBMSs - applications don't need to know whether the data is on disk or in main memory
- *disk block*
  - sequence of contiguous bytes
  - unit for data storage
  - unit for data transfer (reading data from disk / writing data to disk)
  - reading / writing a block - an input / output (I/O) operation
- *tracks*
  - concentric rings containing blocks, recorded on one or more platters

## Secondary Storage – Magnetic Disks

- *sectors*
  - arcs on tracks
- *platters*
  - single-sided, double-sided (data recorded on one / both surfaces)
- *cylinder*
  - set of all tracks with the same diameter
- *disk heads*
  - one per recorded surface
  - to read / write a block, a head must be on top of the block
  - all disk heads are moved as a unit
  - systems with one active head

## Secondary Storage – Magnetic Disks

- sector size
  - characteristic of the disk, cannot be modified
- block size
  - multiple of the sector size



[Si08]

## Secondary Storage – Magnetic Disks

- DBMSs operate on data when it is in memory
- block - unit for data transfer between disk and main memory
- time to access a desired location:
  - main memory - approximately the same for any location
  - disk - depends on where the data is stored
- disk access time:
  - seek time + rotational delay + transfer time
  - seek time
    - time to move the disk head to the desired track (smaller platter size => decreased seek time)
  - rotational delay
    - time for the block to get under the head
  - transfer time
    - time to read / write the block, once the disk head is positioned over it



## Secondary Storage – Magnetic Disks

- time required for DB operations - dominated by the time taken to transfer blocks between disk and main memory
- goal
  - minimize access time
  - for this purpose, data should be carefully placed on disk
- records that are often used together should be close to each other:
  - same block
  - same track
  - same cylinder
  - adjacent cylinder
- accessing data in a sequential fashion reduces seek time and rotational delay

## Secondary Storage – Magnetic Disks

### \* characteristics, e.g.:

- storage capacity (e.g., GB)
- platters
  - number, *single-sided* or *double-sided*
- average / max seek time (ms)
- average rotational delay (ms)
- number of rotations / min
- data transfer rate (MB/s)
- ...

# Moore's Law

- Gordon Moore: "the improvement of integrated circuits is following an exponential curve that doubles every 18 months"
  - parameters that follow Moore's law
    - speed of processors (number of instructions executed / sec)
    - no. of bits / chip
    - capacity of largest disks
  - parameters that do not follow Moore's law
    - speed of accessing data in main memory
    - disk rotation speed
- => "latency" keeps increasing
- time to move data between memory hierarchy levels appears to take longer compared with computation time

## Solid-State Disks

- NAND flash components
- faster random access
- higher data transfer rates
- no moving parts
- higher cost per GB
- limited write cycles

## Managing Disk Space

- the *disk space manager* (DSM) manages space on disk
- page
  - unit of data
  - size of a page = size of a disk block
  - R/W a page - one I/O operation
- upper layers in the DBMS can treat the data as a collection of pages
- DSM
  - commands to allocate / deallocate / read / write a page
  - knows which pages are on which disk blocks
  - monitors disk usage, keeping track of available disk blocks

## Managing Disk Space

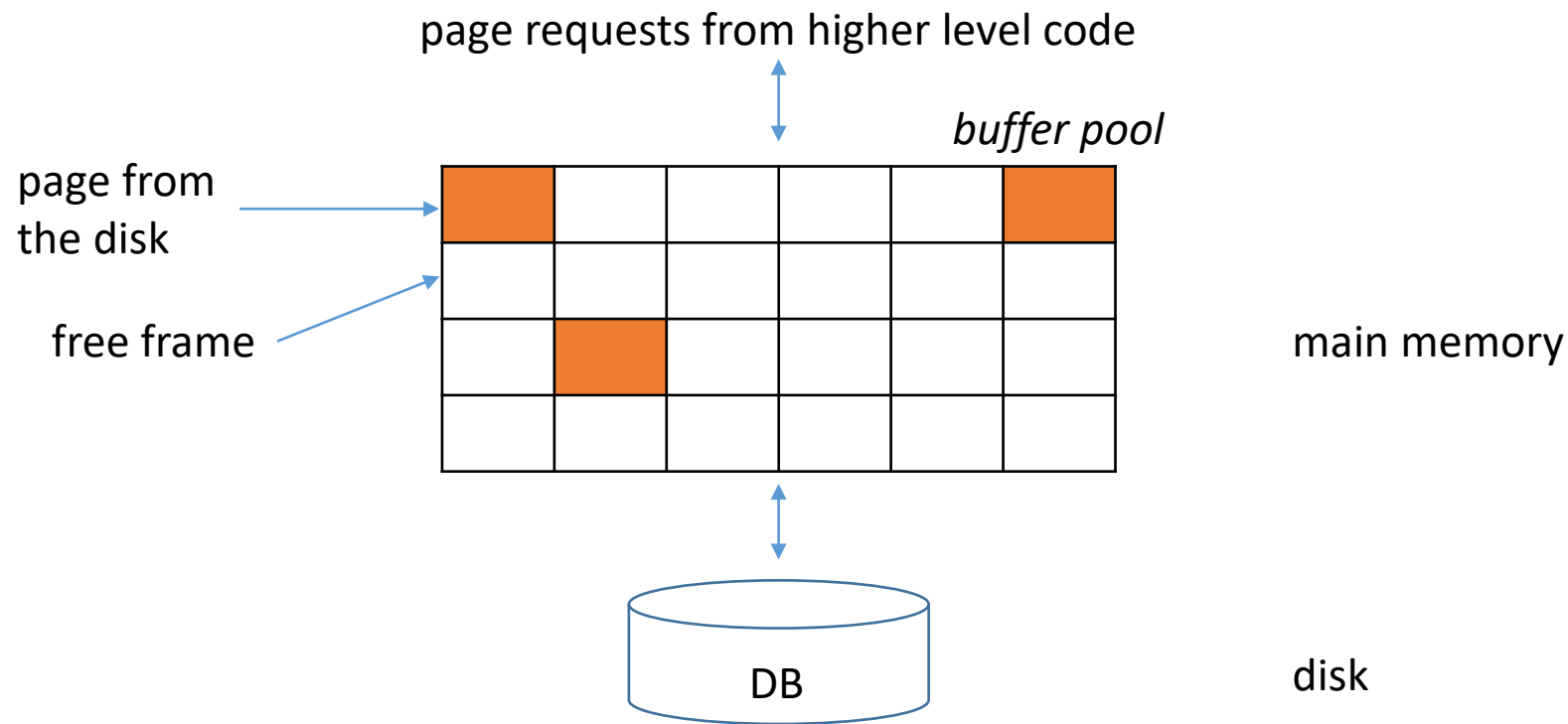
- free blocks can be identified:
  - by maintaining a linked list of free blocks (on deallocation, a block is added to the list)
  - by maintaining a bitmap with one bit / block, indicating whether the corresponding block is used or not
    - allows for fast identification of contiguous available areas on disk

# Buffer Manager

- e.g., DB = 500.000 pages, main memory - 1000 available pages, query that scans the entire file
- *buffer manager* (BM)
  - brings new data pages from disk to main memory as they are required
  - decides what main memory pages can be replaced
  - manages the available main memory
    - collection of pages called the *buffer pool* (BP)
    - *frame*
      - page in the BP
      - slot that can hold a page
- *replacement policy*
  - policy that dictates the choice of replacement frames in the BP

# Buffer Manager

- higher level layer L in the DBMS asks the BM for page P
- if P is not in the BP, the BM brings it into a frame F in the BP
- when P is no longer needed, L notifies the BM (it releases P), so F can be reused
- if P has been modified, L notifies the BM, which propagates the changes in F to the disk





# Buffer Manager

- BM maintains 2 variables for each frame F
  - *pin\_count*
    - number of current users (requested the page in F but haven't released it yet)
    - only frames with  $\text{pin\_count} = 0$  can be chosen as replacement frames
  - *dirty*
    - boolean value indicating whether the page in F has been changed since being brought into F
- incrementing  $\text{pin\_count}$ 
  - pinning a page P in a frame F
- decrementing  $\text{pin\_count}$ 
  - unpinning a page

## Buffer Manager

- initially,  $\text{pin\_count} = 0$ ,  $\text{dirty} = \text{off}$ ,  $\forall F \in \text{BP}$
- L asks for a page P; the BM:
  1. checks whether page P is in the BP; if so,  $\text{pin\_count}(F)++$ , where F is the frame containing Potherwise:
  - a. BM chooses a frame FR for replacement
    - if the BP contains multiple frames with  $\text{pin\_count} = 0$ , one frame is chosen according to the BM's replacement policy
    - $\text{pin\_count}(\text{FR})++$ ;
  - b. if  $\text{dirty}(\text{FR}) = \text{on}$ , BM writes the page in FR to disk
  - c. BM reads page P in frame FR
- 2. the BM returns the address of the BP frame that contains P to L

## Buffer Manager

- obs. if no BP frame has `pin_count = 0` and page P is not in BP, BM has to wait / the transaction may be aborted
- page requested by several transactions; no conflicting updates
- crash recovery, Write-Ahead Log (WAL) protocol - additional restrictions when a frame is chosen for replacement
- replacement policies
  - *Least Recently Used (LRU)*
    - queue of pointers to frames with `pin_count = 0`
    - a frame is added to the end of the queue when its `pin_count` becomes 0
    - the frame at the head of the queue is chosen for replacement
  - *Most Recently Used (MRU)*
  - *random*
  - ...

# Buffer Manager

- replacement policies
  - *clock replacement*
    - LRU variant
    - $n$  – number of frames in BP
    - frame - *referenced* bit; set to *on* when *pin\_count* becomes 0
    - *crt* variable - frames 1 through  $n$ , circular order
    - if the current frame is not chosen, then  $crt++$ , examine next frame
  - if  $pin\_count > 0$ 
    - current frame not a candidate,  $crt++$
  - if *referenced* = *on*
    - *referenced* := *off*,  $crt++$
  - if  $pin\_count = 0$  AND *referenced* = *off*
    - choose current frame for replacement

# Buffer Manager

- replacement policies
  - can have a significant impact on performance
- example:
  - BM uses LRU
  - repeated scans of file  $f$
  - BP: 5 frames,  $f$ :  $\leq 5$  pages
    - first scan of  $f$  brings all the pages in the BP
    - subsequent scans find all the pages in the BP
  - BP: 5 frames,  $f$ : 6 pages
    - *sequential flooding*: every scan of  $f$  reads all the pages
    - MRU – better in this case

# Disk Space Manager & Buffer Manager

- DSM
  - portability - different OSs
- BM
  - DBMS can anticipate the next several page requests (operations with a known page access pattern, like sequential scans)
  - *prefetching* - BM brings pages in the BP before they are requested
  - prefetched pages
    - contiguous: faster reading (than reading the same pages at different times)
    - not contiguous: determine an access order that minimizes seek times / rotational delays

# Disk Space Manager & Buffer Manager

- BM
  - DBMS needs
    - ability to explicitly force a page to disk
    - ability to write some pages to disk before other pages are written
      - WAL protocol - first write log records describing page changes, then write modified page

# References

- [Ta13] ȚÂMBULEA, L., Curs Baze de date, Facultatea de Matematică și Informatică, UBB, 2013-2014
- [Ra02] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (3rd Edition), McGraw-Hill, 2002
- [Da03] DATE, C.J., An Introduction to Database Systems (8<sup>th</sup> Edition), Addison-Wesley, 2003
- [Ga09] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book (2nd Edition), Pearson Education, 2009
- [Ha96] HANSEN, G., HANSEN, J., Database Management And Design (2<sup>nd</sup> Edition), Prentice Hall, 1996
- [Ra02S] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, Slides for the 3<sup>rd</sup> Edition,  
<http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html>
- [Si11] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts (6th Edition), McGraw-Hill, 2011
- [Si19S] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts, Slides for the 7th Edition, <http://codex.cs.yale.edu/avi/db-book/>
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems,  
<http://infolab.stanford.edu/~ullman/fcdb.html>