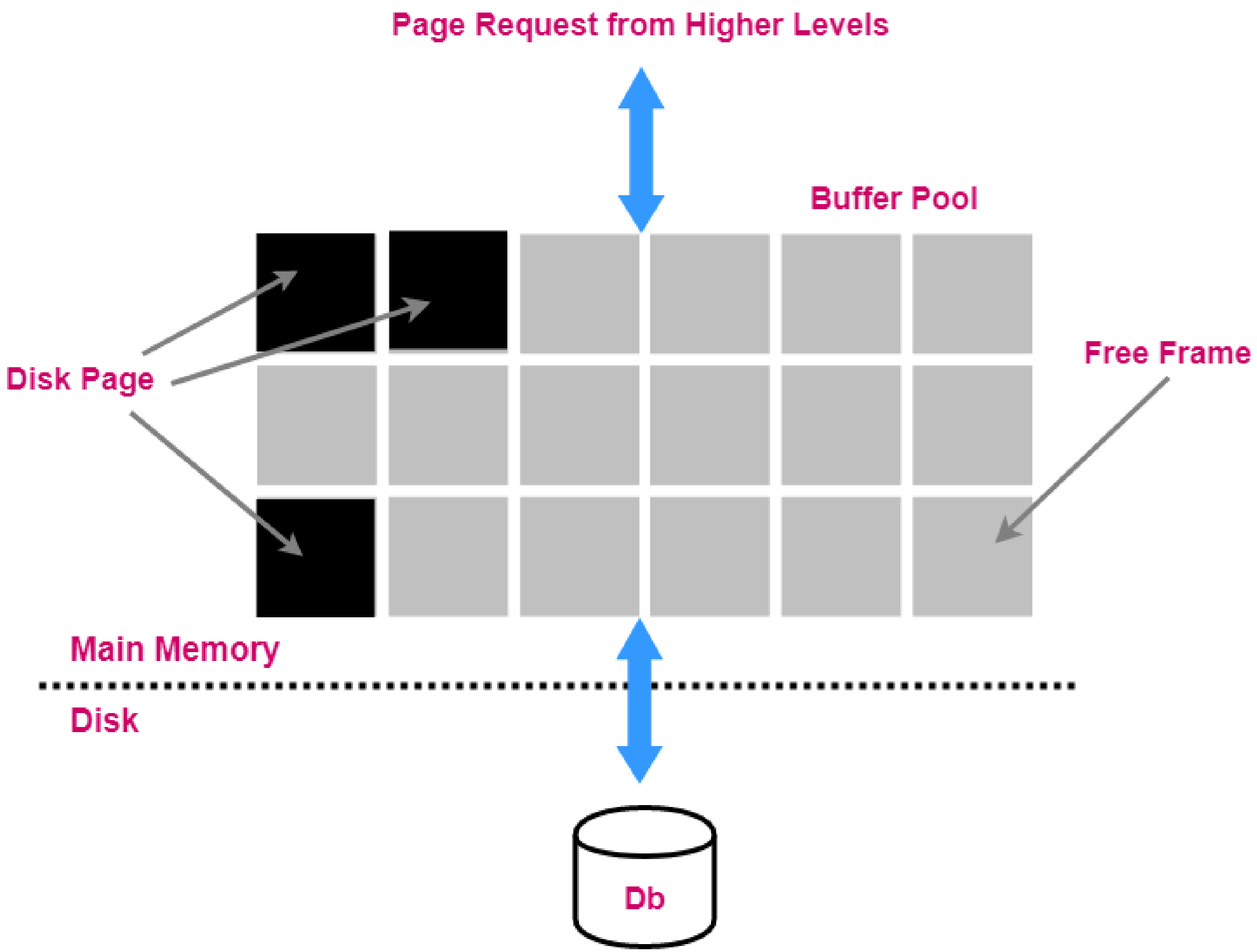
Buffer Management

Project #2

Slides based on "Database Management Systems" book by Johannes Gehrke and Raghu Ramakrishnan

Buffer Management in a DBMS



- Data must be in memory for DBMS to operate on it
- Table of <*frame#*, *pageid*> pair is maintained.
- Bookkeeping information (per frame):
  - *pin count*
  - *dirty bit*
- Choice of the frame is dictated by **replacement policy**.

When a Page is requested …

- If requested page is not in pool and the pool is full:
  - Choose a frame for *replacement*.
  - If frame is dirty, write it to disk.
  - Read requested page into chosen frame.
- *Pin* the page and return its address.

- If request can be predicted (e.g., sequential scans), pages can be *pre-fetched* (several pages at the same time)

More on Buffer Management

- Requestor of page must unpin it and indicate whether page has been modified:
  - *dirty* bit is used for this
- Page in pool may be requested many times:
  - A *pin count* is used.
  - A page is candidate for replacement iff *pin count == 0*.
- CC & Recovery may entail additional I/O when frame is chosen for replacement (*Write-Ahead Log* Protocol).

Buffer Replacement Policy

- Frame is chosen for replacement by a *replacement policy*:
  - FIFO, Least-Recently-Used (LRU), Clock, MRU, etc.
- Policy can have a big impact on # of I/O's; depends on the *access pattern*.
- Sequential Flooding. Nasty situation caused by LRU + repeated sequential scans.
  - *# buffer frames < # pages in file*

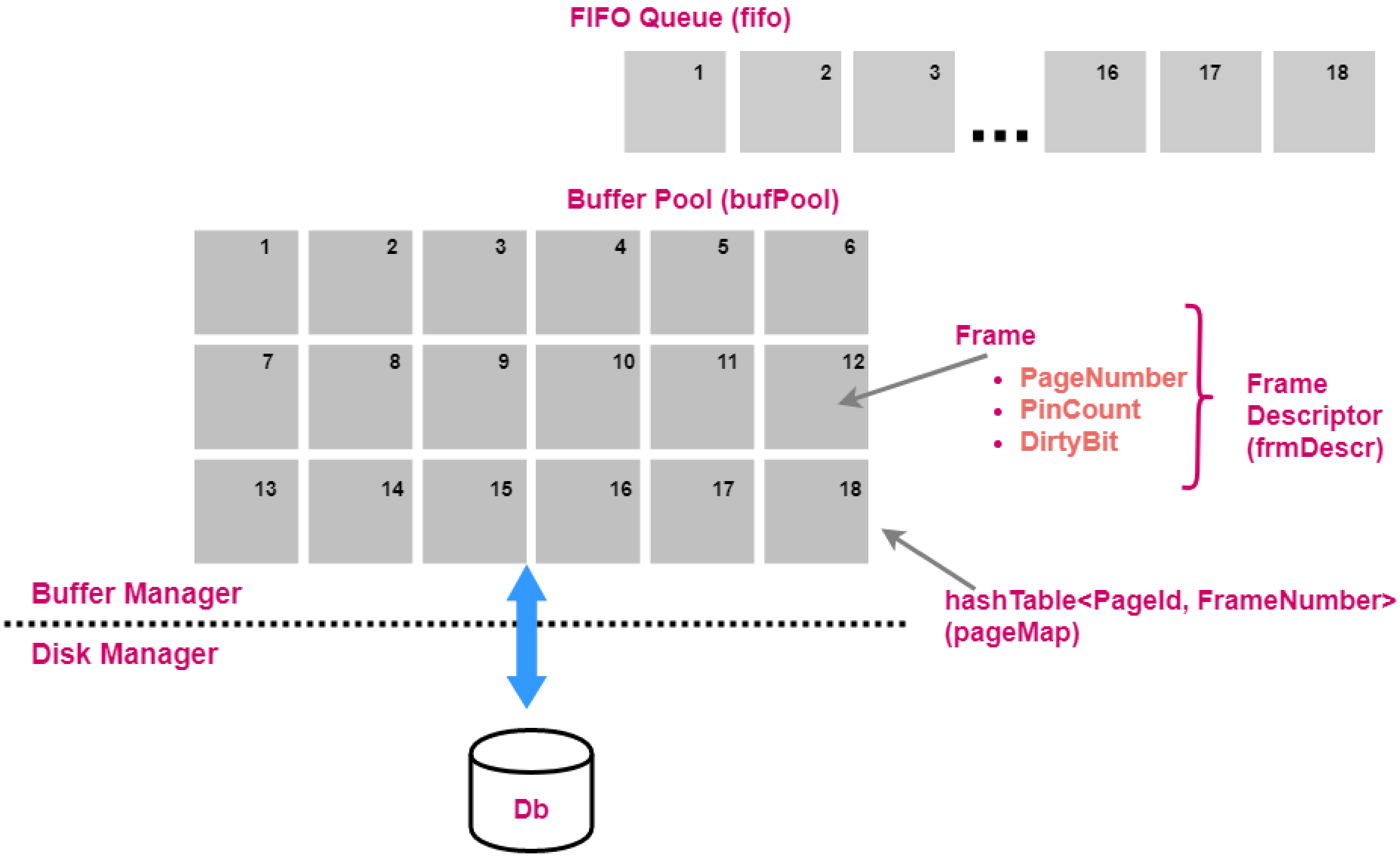
    means each page request causes an I/O. MRU much better in this situation (but not in all situations, of course).

DBMS vs OS File System

OS does disk space & buffer management:
why not let OS manage these tasks?

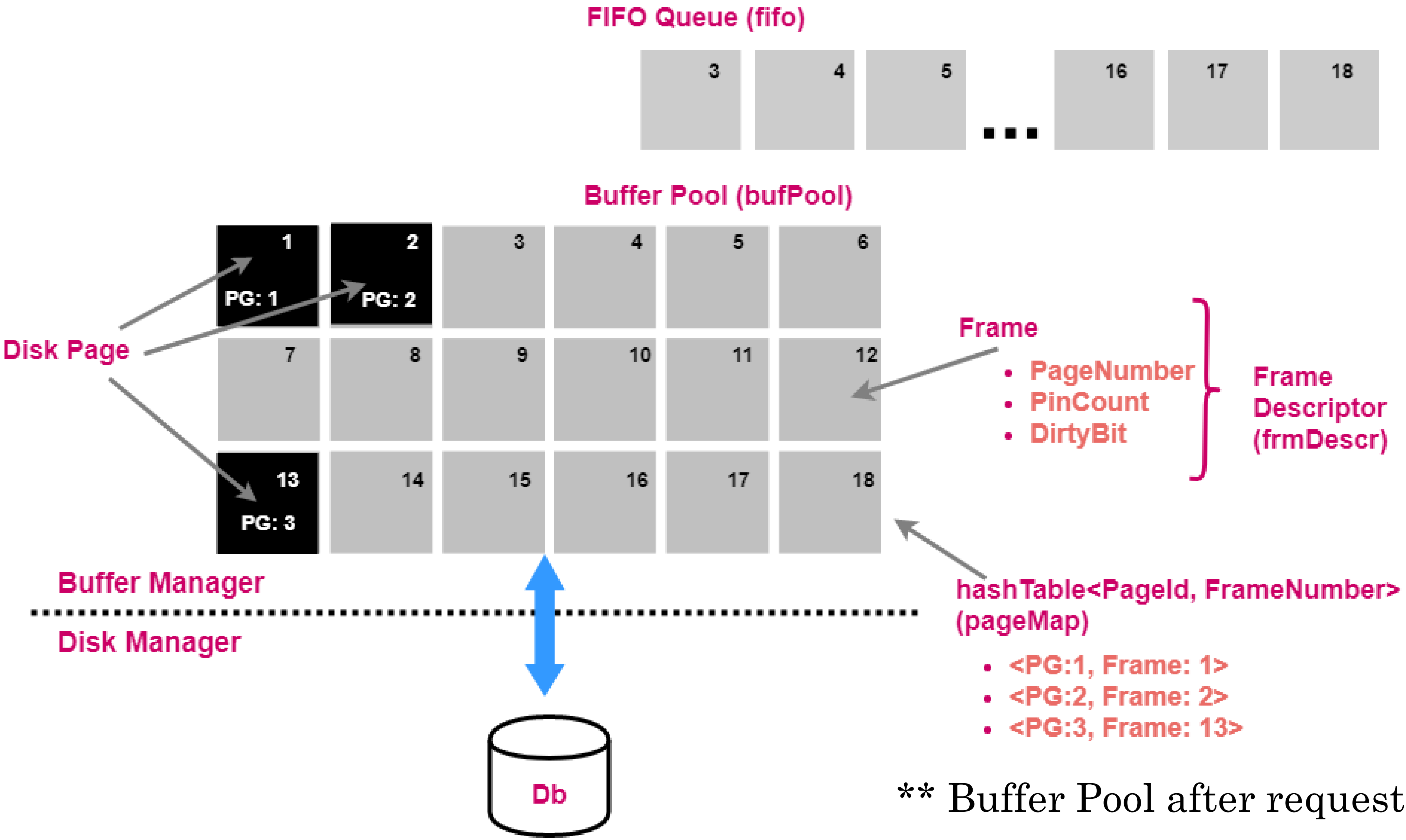
- Differences in OS support: portability issues
- Some limitations, e.g., files can't span disks.
- Buffer management in DBMS requires ability to:
  - *pin a page* in buffer pool, *force a page* to disk (important for implementing CC & recovery),
  - adjust *replacement policy*, and prefetch pages based on access patterns in typical DB operations.

Project #2



FIFO Queue (fifo)
1
2
3
...
16
17
18
Buffer Pool (bufPool)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
Frame
PageNumber
PinCount
DirtyBit
Frame Descriptor (frmDescr)
hashTable<PageId, FrameNumber> (pageMap)
Buffer Manager
Disk Manager
Db

Project #2



FIFO Queue (fifo)

Buffer Pool (bufPool)

Disk Page

Frame

- PageNumber
- PinCount
- DirtyBit

Frame Descriptor (frmDescr)

hashTable<PageId, FrameNumber> (pageMap)

- <PG:1, Frame: 1>
- <PG:2, Frame: 2>
- <PG:3, Frame: 13>

Buffer Manager

Disk Manager

** Buffer Pool after requesting 3 pages.

Methods to Implement

- ***void pinPage***

  Attempts to pin the requested page.

- ***void unpinPage***

  Attempts to unpin the requested page.

- ***PageId newPage***

  Attempts to allocate ‘x’ pages in memory.

- ***void freePage***

  Attempts to the case when we need to remove a page completely from disk.