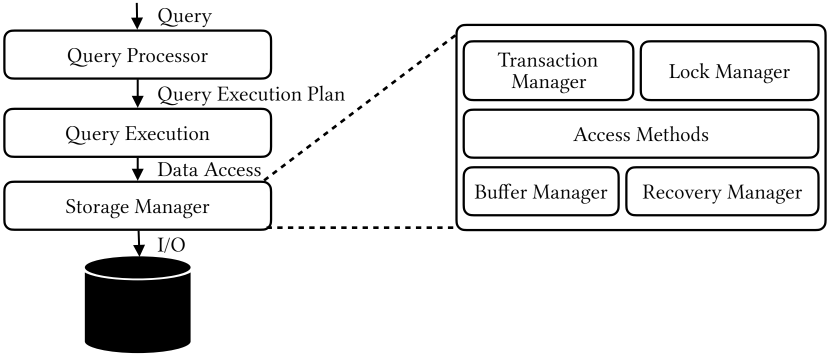
COL362/632 Introduction to Database Management Systems

Database Systems – Buffer Manager

Kaustubh Beedkar

Department of Computer Science and Engineering
Indian Institute of Technology Delhi

Database Storage

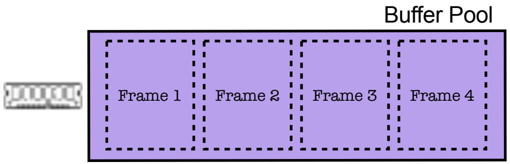


- DBMS stores database in files on disk
- Buffer pool manager takes care of memory management and Disk $\xleftrightarrow{Data}$ Memory

Buffer Manager

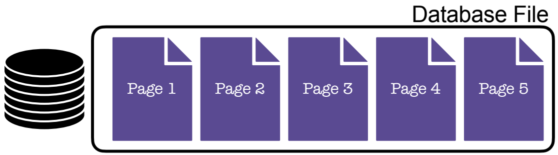
Buffer Pool

- An **array** of fixed-size page in the memory
- Entry in the array is called a **frame**



Buffer Manager

- Copies a page from (to) disk to (from) buffer pool



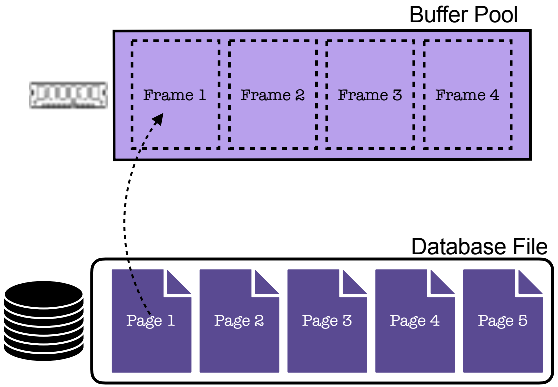
Buffer Manager

Buffer Pool

- An **array** of fixed-size page in the memory
- Entry in the array is called a **frame**

Buffer Manager

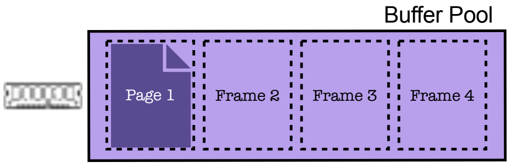
- Copies a page from (to) disk to (from) buffer pool



Buffer Manager

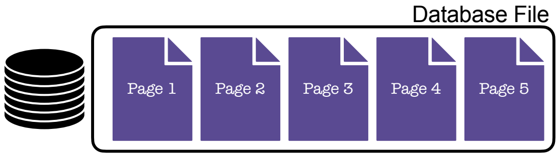
Buffer Pool

- An **array** of fixed-size page in the memory
- Entry in the array is called a **frame**



Buffer Manager

- Copies a page from (to) disk to (from) buffer pool



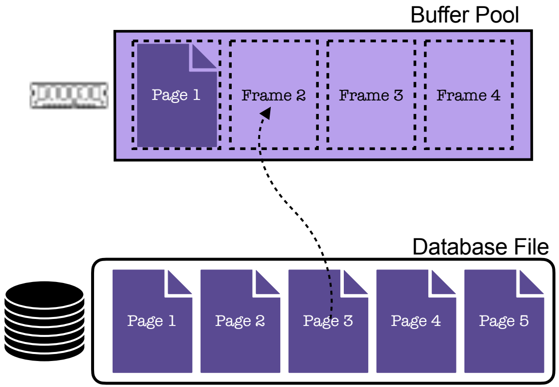
Buffer Manager

Buffer Pool

- An **array** of fixed-size page in the memory
- Entry in the array is called a **frame**

Buffer Manager

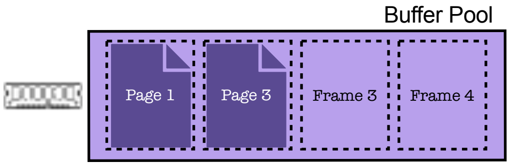
- Copies a page from (to) disk to (from) buffer pool



Buffer Manager

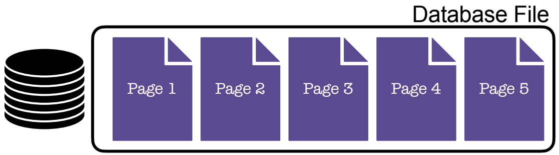
Buffer Pool

- An **array** of fixed-size page in the memory
- Entry in the array is called a **frame**



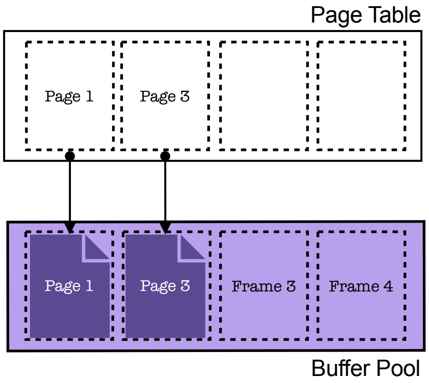
Buffer Manager

- Copies a page from (to) disk to (from) buffer pool



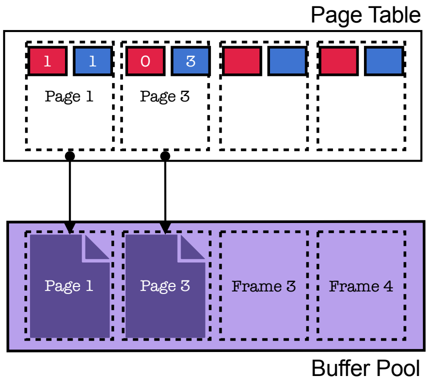
Page Table

- In-memory data structure
- Tracks mappings of pages in memory (**note:** this is different from the page directory)
- Stores additional metadata per page
  - **dirty bit**
    - Whether or not the page has been modified in memory
  - **pin count**
    - Number of current queries using the page



Page Table

- In-memory data structure
- Tracks mappings of pages in memory (**note:** this is different from the page directory)
- Stores additional metadata per page
    - **dirty bit**
        - Whether or not the page has been modified in memory
    - **pin count**
        - Number of current queries using the page



Page Requests

- **Read (page)**
  Read page from disk into buffer pool
  (if not already present)

- **Flush (page)**
  Evict page from buffer pool
  and write to disk

- **Release (page)**
  Evict page from buffer pool
  without writing to disk

Page Requests

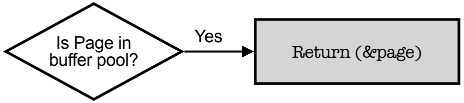
- **Read (page)**
  Read page from disk into buffer pool
  (if not already present)
- **Flush (page)**
  Evict page from buffer pool
  and write to disk
- **Release (page)**
  Evict page from buffer pool
  without writing to disk

Page Requests

- **Read (page)**
  Read page from disk into buffer pool
  (if not already present)

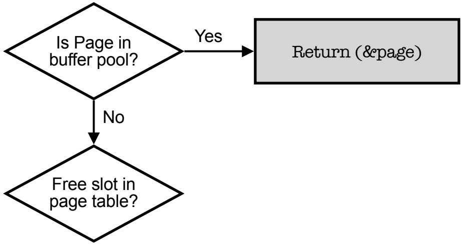
- **Flush (page)**
  Evict page from buffer pool
  and write to disk

- **Release (page)**
  Evict page from buffer pool
  without writing to disk



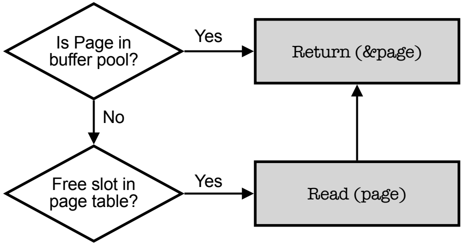
Page Requests

- **Read (page)**
  Read page from disk into buffer pool
  (if not already present)
- **Flush (page)**
  Evict page from buffer pool
  and write to disk
- **Release (page)**
  Evict page from buffer pool
  without writing to disk



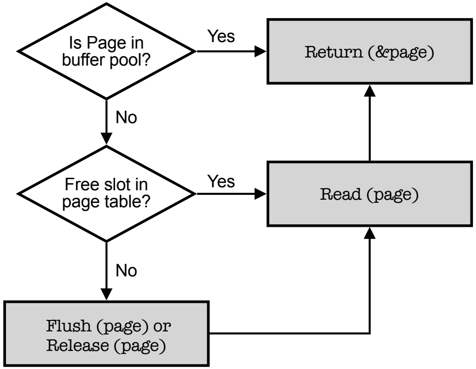
Page Requests

- **Read (page)**
  Read page from disk into buffer pool (if not already present)
- **Flush (page)**
  Evict page from buffer pool and write to disk
- **Release (page)**
  Evict page from buffer pool without writing to disk



Page Requests

- **Read (page)**
  Read page from disk into buffer pool (if not already present)
- **Flush (page)**
  Evict page from buffer pool and write to disk
- **Release (page)**
  Evict page from buffer pool without writing to disk

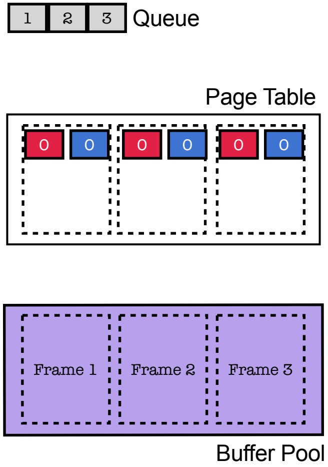


Buffer Replacement Policy

- Which page to evict from the buffer pool?
- Policies
  - **L**east-**R**ecently **U**sed (LRU)
  - **M**ost-**R**ecently **U**sed (MRU)
  - Clock
  - First-in First-out (FIFO)
  - Random
  - . . .
- Choice of policy impacts speed, accuracy, and correctness

LRU

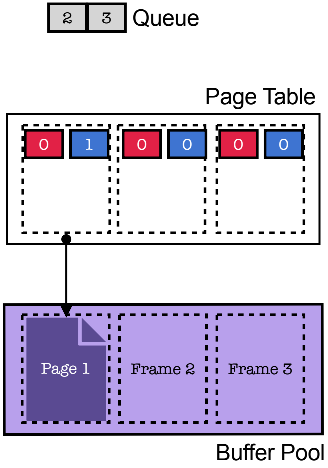
Example


1
2
3
Queue
Page Table
0
0
0
0
0
0
Frame 1
Frame 2
Frame 3
Buffer Pool

LRU

Example

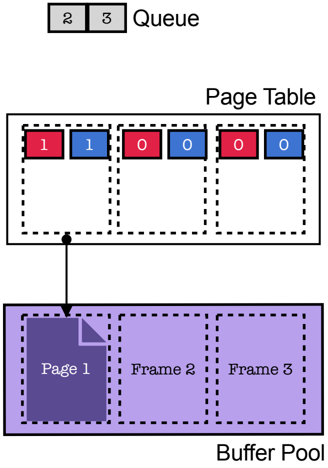
1. `request Page 1`



LRU

Example

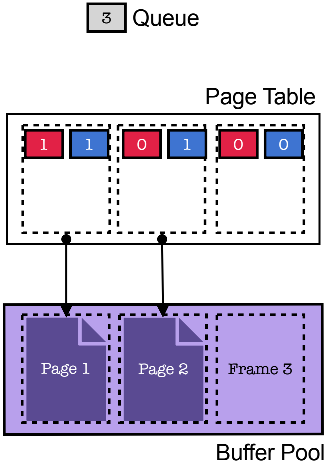
1. `request  Page 1`
2. `modify   Page 1`



LRU

Example

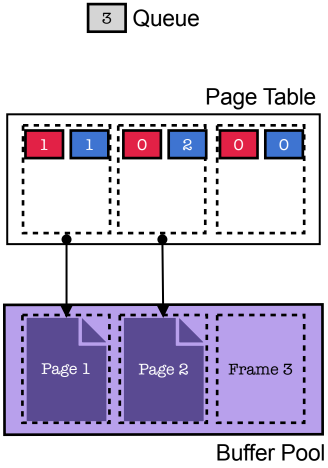
1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`



LRU

Example

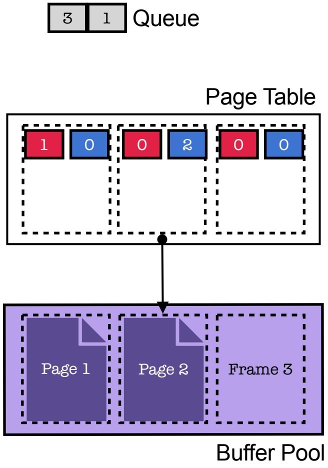
1. `request  Page 1`
2. `modify   Page 1`
3. `request  Page 2`
4. `request  Page 2`



LRU

Example

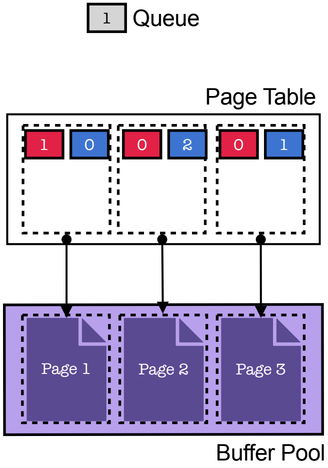
1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`



LRU

Example

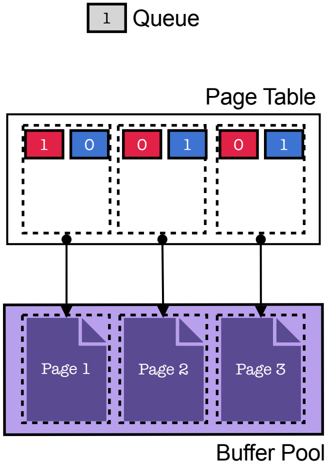
1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`



LRU

Example

1. `request Page 1`
2. `modify Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`
7. `release Page 2`

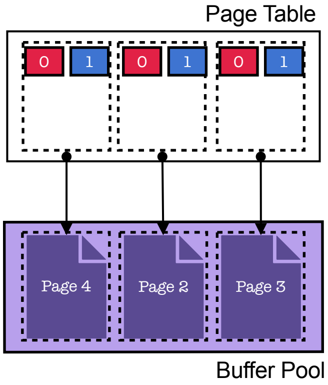


LRU

Example

1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`
7. `release Page 2`
8. `request Page 4`

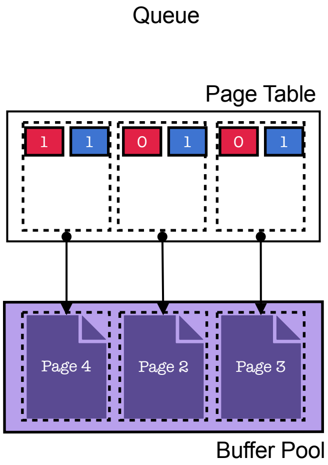
Queue



LRU

Example

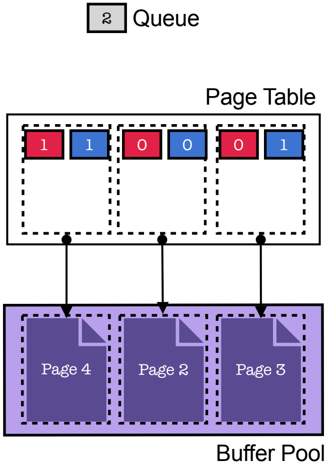
1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`
7. `release Page 2`
8. `request Page 4`
9. `modify  Page 4`



LRU

Example

1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`
7. `release Page 2`
8. `request Page 4`
9. `modify  Page 4`
10. `release Page 2`

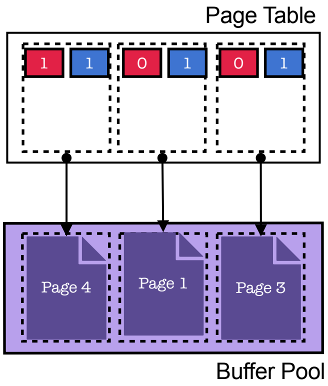


LRU

Example

1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`
7. `release Page 2`
8. `request Page 4`
9. `modify  Page 4`
10. `release Page 2`
11. `request Page 1`

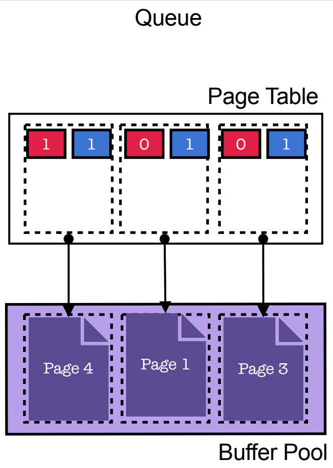
Queue



LRU

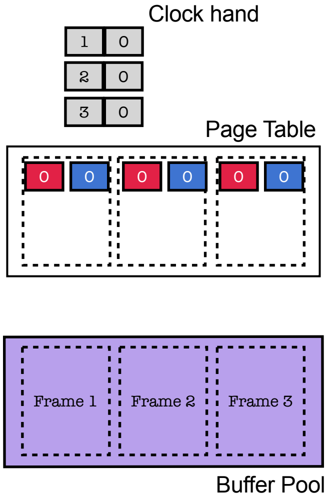
Example

1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`
7. `release Page 2`
8. `request Page 4`
9. `modify  Page 4`
10. `release Page 2`
11. `request Page 1`
12. `request Page 5`



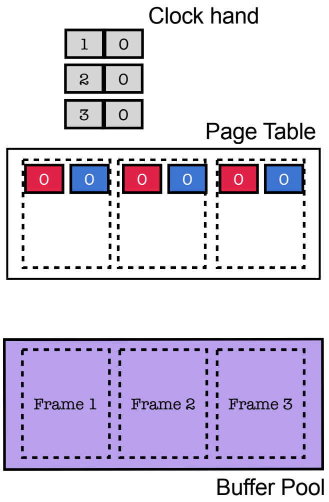
CLOCK

- LRU variant with lower memory
- Frames as organized as circular buffer
- Maintains a reference bit for each frame
  - Set to 1 when pin count = 0
- Pointer to frame (clock hand)
- if pin count > 0, increment pointer
- if ref. bit = 1, reset and increment pointer
- if ref. bit = 0, and pint count = 0, select the page



CLOCK

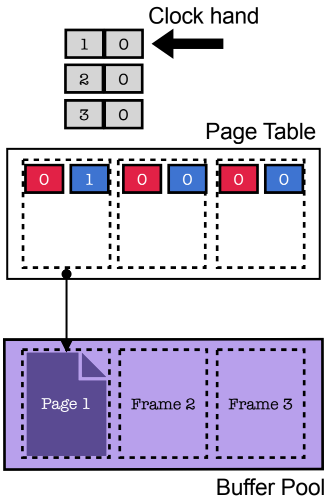
Example



CLOCK

Example

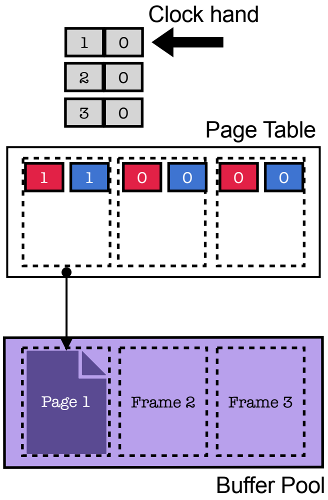
1. `request Page 1`



CLOCK

Example

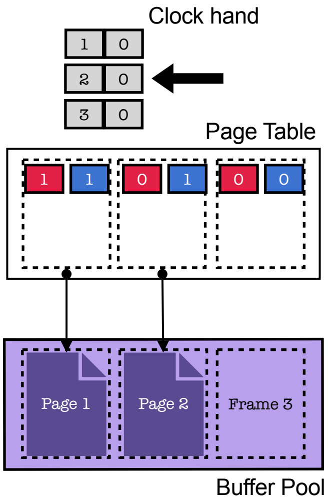
1. `request Page 1`
2. `modify  Page 1`



CLOCK

Example

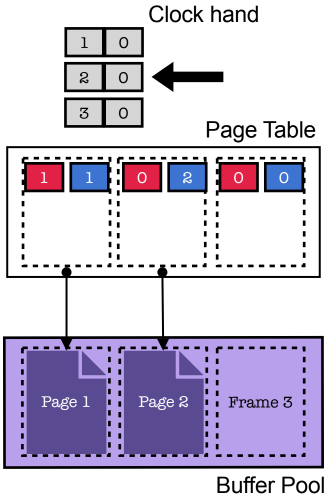
1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`



CLOCK

Example

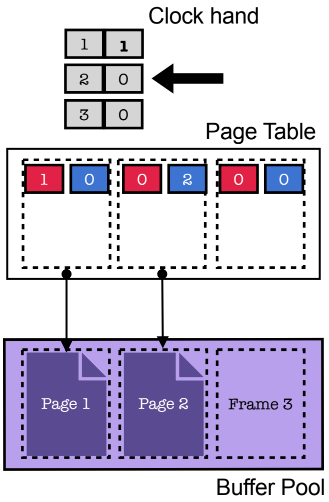
1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`



CLOCK

Example

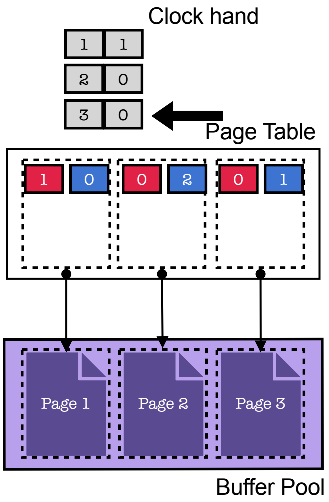
1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`



CLOCK

Example

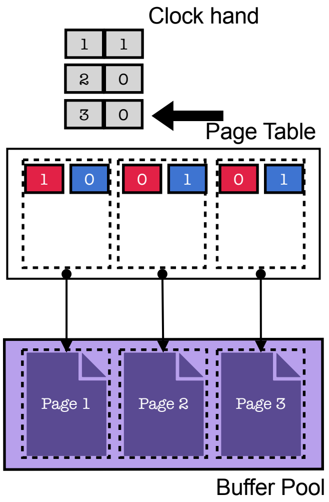
1. `request  Page 1`
2. `modify   Page 1`
3. `request  Page 2`
4. `request  Page 2`
5. `release  Page 1`
6. `request  Page 3`



CLOCK

Example

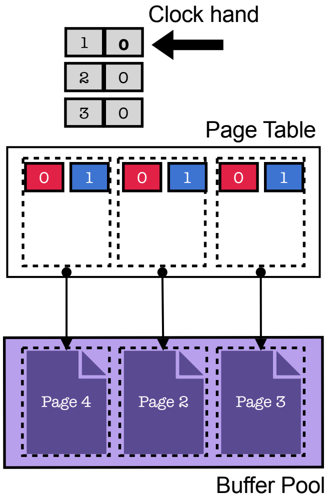
1. `request Page 1`
2. `modify Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`
7. `release Page 2`



CLOCK

Example

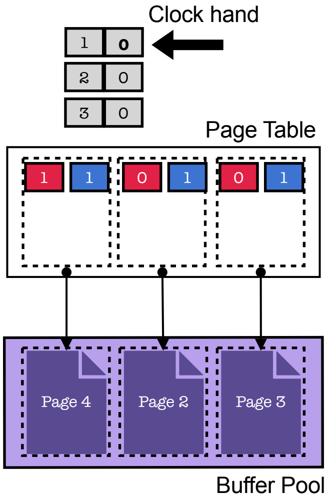
1. `request   Page 1`
2. `modify    Page 1`
3. `request   Page 2`
4. `request   Page 2`
5. `release   Page 1`
6. `request   Page 3`
7. `release   Page 2`
8. `request   Page 4`



CLOCK

Example

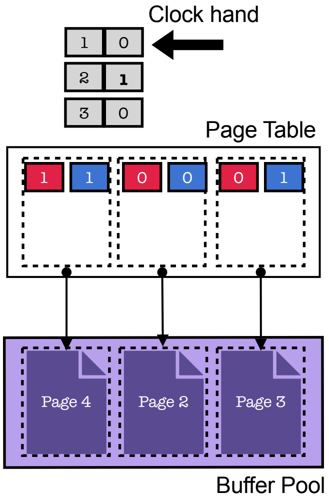
1. `request  Page 1`
2. `modify   Page 1`
3. `request  Page 2`
4. `request  Page 2`
5. `release  Page 1`
6. `request  Page 3`
7. `release  Page 2`
8. `request  Page 4`
9. `modify   Page 4`



CLOCK

Example

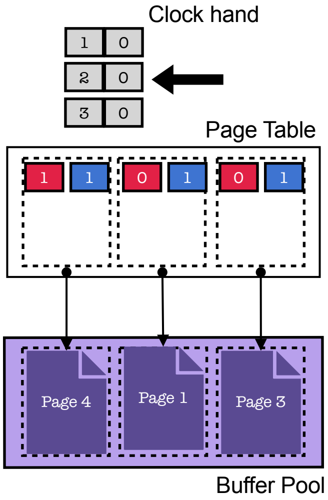
1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`
7. `release Page 2`
8. `request Page 4`
9. `modify  Page 4`
10. `release Page 2`



CLOCK

Example

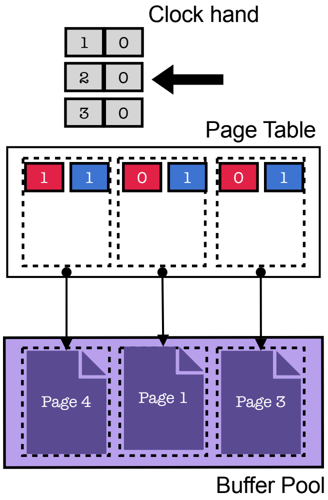
1. `request  Page 1`
2. `modify   Page 1`
3. `request  Page 2`
4. `request  Page 2`
5. `release  Page 1`
6. `request  Page 3`
7. `release  Page 2`
8. `request  Page 4`
9. `modify   Page 4`
10. `release  Page 2`
11. `request  Page 1`



CLOCK

Example

1. `request Page 1`
2. `modify  Page 1`
3. `request Page 2`
4. `request Page 2`
5. `release Page 1`
6. `request Page 3`
7. `release Page 2`
8. `request Page 4`
9. `modify  Page 4`
10. `release Page 2`
11. `request Page 1`
12. `request Page 5`



Sequential Flooding

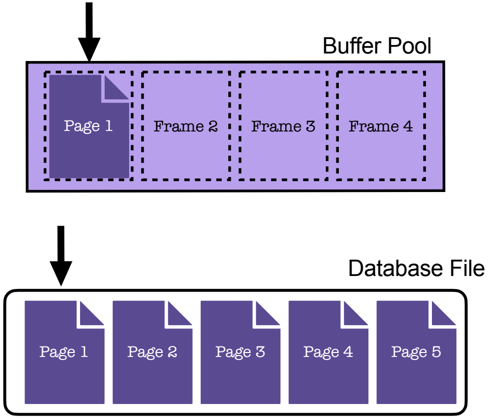
- `select * from table where table.col = val`
  - Sequential Scan
- LRU and CLOCK can lead to **sequential flooding**
- Pages are only read once, and not used again
- MRU is preferable over LRU for repeated sequential scans

Optimizations

- **Pre-fetching pages**

  e.g., based on the query plan ('ll be covered in later lectures)

  - **Sequential Scans**
  - Index Scans

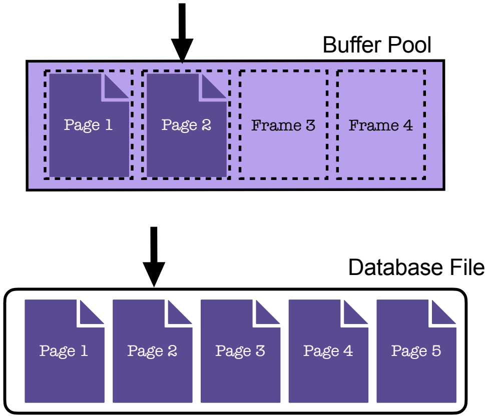


Optimizations

- **Pre-fetching pages**

  e.g., based on the query plan ('ll be covered in later lectures)

  - **Sequential Scans**
  - Index Scans

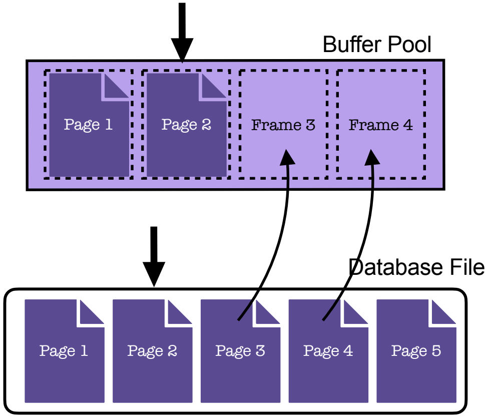


Optimizations

- **Pre-fetching pages**

  e.g., based on the query plan ('ll be covered in later lectures)

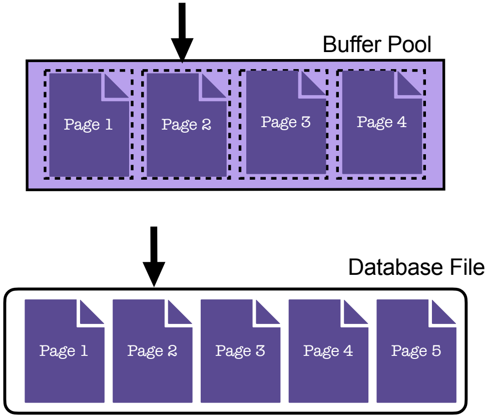
  - **Sequential Scans**
  - Index Scans



Optimizations

- **Pre-fetching pages**

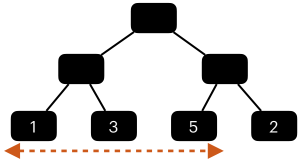
  e.g., based on the query plan ('ll be covered in later lectures)
  - **Sequential Scans**
  - Index Scans

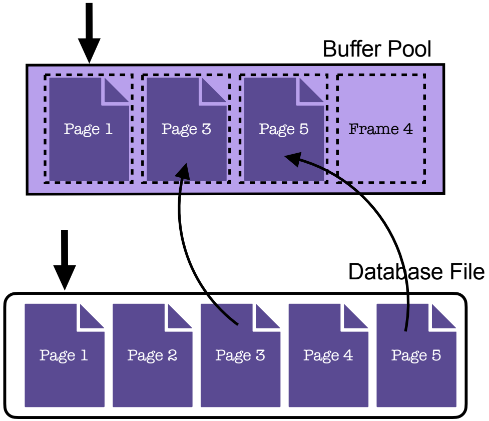


Optimizations

- **Pre-fetching pages**

  e.g., based on the query plan ('ll be covered in later lectures)
  - Sequential Scans
  - **Index Scans**





Other Aspects

- Optimization
  - Scan sharing
- Bypassing OS page cache
- DBMS may have more than one buffer pool
  - Per page
  - Per database
  - Multiple instances
- Buffer pool management
  - Global policies — Decisions based on all running queries
  - Local policies — Decisions only based on specific query