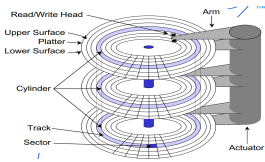# L1 - Data Storage

Magnetic Disks

- **Disk Access Time** Seek time + Rotational Latency + Transfer time
- **Response time** Queueing delay + Disk access time
- **Rotational Delay** $\frac{1}{2} \frac{60s}{RPM}$
- **Transfer Time** sectors on the same track * $\frac{TimePerRevolution}{SectorsPerTrack}$

Buffer Manager

- **Buffer pool** Main memory allocated for DBMS
- **pin count** is incremented upon pinning
- **dirty bit** is updated when the page is unpinned (if modified)
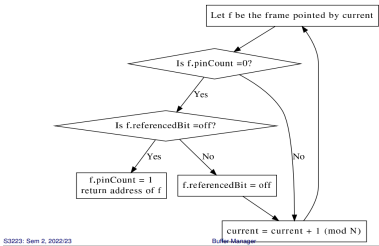- **R**eplacement is only possbile if pin count == 0
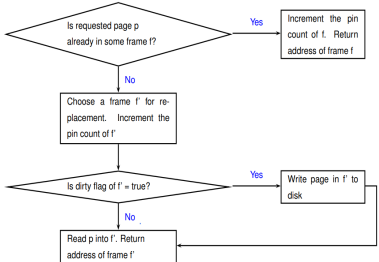
Replacement Policies

LRU Policy
- Maintains a queue of pointers to frames with pin count = 0
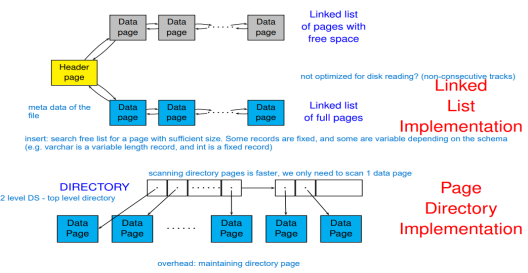
Clock Replacement Policy
N = number of frames in buffer pool

- Simplifies LRU with a second chance round robin system
- Each frame has a **reference bit** that is turned on when pin count reaches 0
- Repalces a page when referenced bit if off and pin count is 0
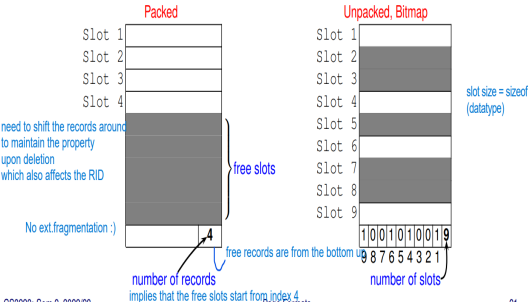

File Organisation

Heap File Implementations

worst case: O(n), and we need to scan through the entire data page to find out the space available

Linked list of pages with free space

not optimized for disk reading? (non-cpnsecutive tracks)

Linked List Implementation

meta data of the file

Linked list of full pages

insert: search free list for a page with sufficient size. Some records are fixed, and some are variable depending on the schema (e.g. varchar is a variable length record, and int is a fixed record)

scanning directory pages is faster, we only need to scan 1 data page

DIRECTORY
2 level DS - top level directory

Page Directory Implementation
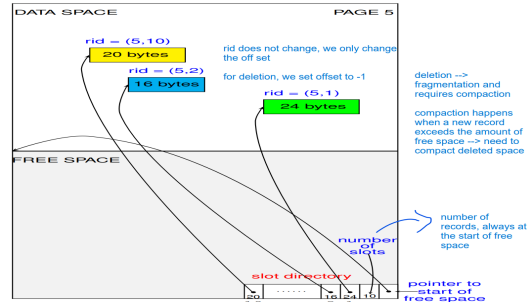
overhead: maintaining directory page

Page Formats: Fixed Length Records
- **Packed Organisation** Store records in contiguous slots
- **Unpacked Organisation** Uses a bit array to maintain free slots

need to shift the records around to maintain the property upon deletion which also affects the RID

No ext.fragmentation :)

number of records

implies that the free slots start from index 4

slot size = sizeof (datatype)

free records are from the bottom up

number of slots

Page Formats: Variable Length Records

- **Slotted Page** Store records in slots of ¡record offset, record length¿
- **Record Offset:** Offset of the record from the start of the page

rid does not change, we only change the off set

for deletion, we set offset to -1

deletion --> fragmentation and requires compaction

compaction happens when a new record exceeds the amount of free space --> need to compact deleted space

number of records, always at the start of free space

pointer to start of free space

Record Formats
▶ Fixed-Length Records
  ▶ Fields are stored consecutively

F1	F2	F3	F4

▶ Variable-Length Records
  ▶ Delimit fields with special symbols

F1	$	F2	$	F3	$	F4

  ▶ Use an array of field offsets

o1	o2	o3	o4	F1	F2	F3	F4

- Each o_i is an offset to beginning of field Fi