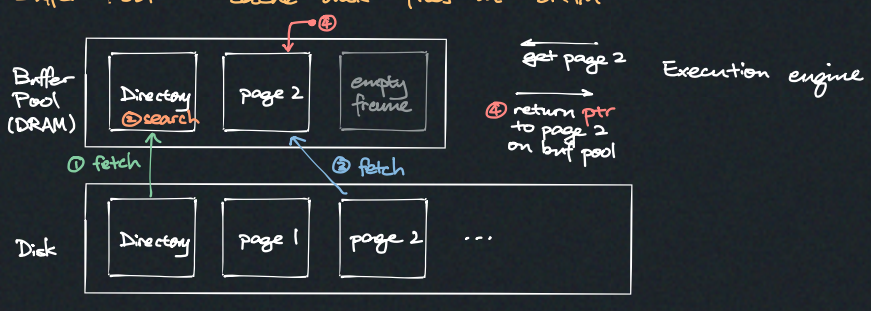
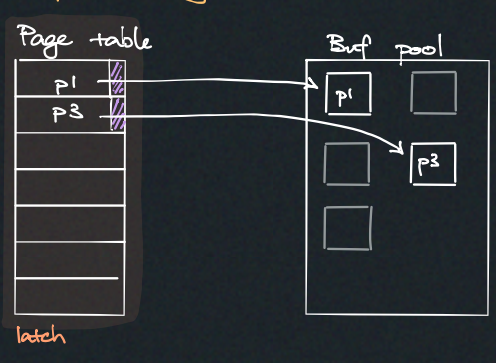
Lec 6 DBMS Memory Management

Buffer Pool — cache disk files in DRAM



- Could bring requested page into any frame
- Is pinned initially, execution engine responsible for telling buf pool when to unpin

Buf Pool Org



Metadata:
- Pin flag
- Dirty flag
- Access tracking
- Pin/ref count

- Writes are kept dirty on buf pool

Locks
protect things on disk when multiple transactions
need rollback support

Latches
protect things in mem when multiple threads
not require rollback support
can be mutex

Great Ideas

▷ Multi Buf Pool

Possible:
- global buf pool
- multi buf pool
- per-page type buf pool
- per-db buf pool

Managing multiple pools
└ one more level of mapping
- use ⟨obj_id, page_id, slot_num⟩
  ↓ decide which pool to use
- hash + load balancing

▷ Pre-fetching

→ If doing sequential scan, pre-fetch subsequent pages
→ Or, if pages in tree structure and scanning leaf nodes, figure out leaf traversal

▷ Scan Sharing

→ If multiple queries scan same place, group them together to reduce IO

Ex.
```
select sum(val) from A;
select avg(val) from A;
```
} try have them read same pages at same time rather than fighting on the buf pool

→ Continuous Scan Sharing
└ Convert table to stream, keep scanning & let queries hop on

→ Light Scan aka Buffer Pool Bypass
└ Scan things on disk but not put in buf pool
mem is local to query needing to scan

Replacement Policy

— correct, high hit rate, fast, minimal metadata

▷ LRU (Least Recently Used)

→ evict oldest access
→ keep sorted struct by timestamp

▷ Clock (faster approx of LRU)

→ put pages in circle, with ref count (not same as pin count, ref count estimates # recent access, most sys sets max ref count to 1)
→ when evicting, find first evictable clockwise from clock hand, while decreasing ref count of things it skips
└ ref count 0 & pin count 0

* Problems with LRU & Clock:
- Sequential flooding
  └ if query scans pages sequential, evic. policy thinks all those pages are recent, even if they won't be used again
  └ also bad if inner loop of query keeps scanning
- Doesn't track access frequency

▷ LRU-K — most modern sys use LRU-2

→ Keep track of k last access
└ balances recency & frequency
→ Corner case: track recently evicted pages so we bring back their history when caching the page again

▷ MySQL approximate LRU-K

→ keep young & old list, move things around to estimate freq

▷ Localisation

→ Keep track of eviction info per-query
└ eg. query saying whether a page is important

Dirty Pages

If evicting read-only page — do nothing

If evicting dirty page — need to write to disk somehow
(may be in-place write... or make new version & chain versions...)

Disk IO Scheduling

We don't want OS to manage this, DB knows better optimisation.

Note: O_DIRECT flag bypass OS cache and just write the bytes

→ background task can do eviction, rather than blocking foreground