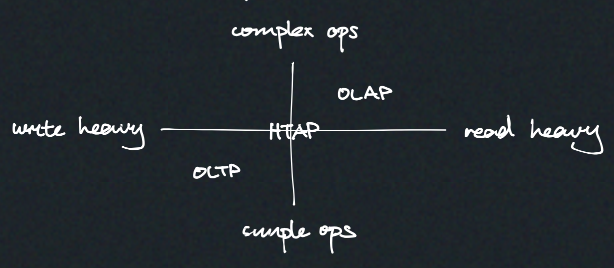
Lec 5 Storage Model & Compression

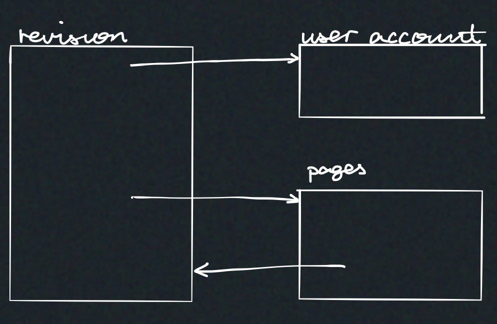
Recall: log-structured vs. index-organised storage

DB workloads

- On-Line Transaction Processing (OLTP)
  fast operations on small amount of data
- On-Line Analytical Processing (OLAP)
  look at all records & produce some report
  or maybe feature engineering in ML
- Hybrid — OLTP and OLAP on same DB (HTAP)
  e.g. ecommerce dealing with carts (OLTP) and ML for product suggestion (OLAP)
  └ sometimes data periodically copied to large OLAP



Ex on Wikipedia



OLTP
- Find single page
- Make one revision
- Log in

OLAP
- See if .gov ppl manipulate wikipedia pages before elections

Storage Models

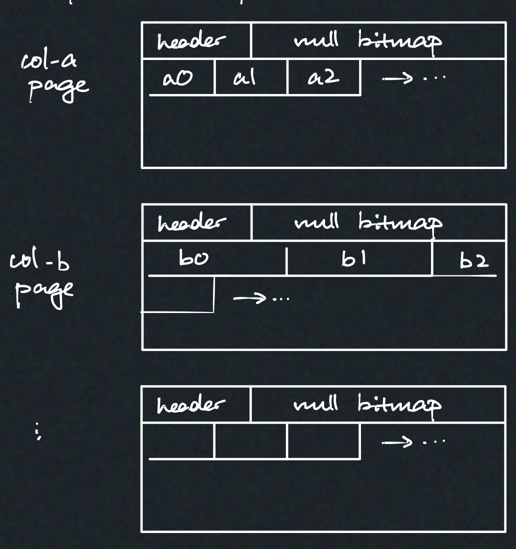
How to physically organise tuples on disk & mem

▷ N-ary Storage Model (NSM) | row ≡ tuple ≡ record
└ contiguous in single slotted page (row-store)

Problem:
- fetching column requires getting entire row
  └ in OLAP query about a single column, a lot of skipping — wasted IOs
- bad space locality
- hard to do col-oriented compression

Good:
- fast insert, delete
- select * fast
- for OLTP

▷ Decomposition Storage Model (DSM)
└ column-store
└ seperate file per attribute



Good
- OLAP
- Per-col compression

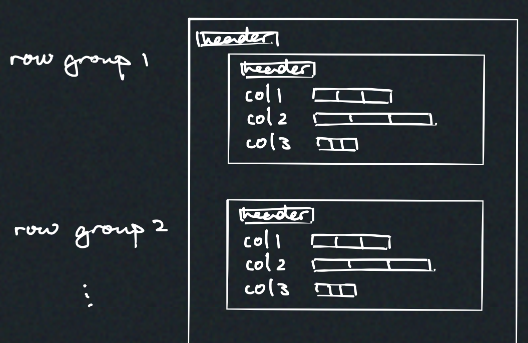
Bad
- select *
- OLTP
- Using many cols blows up buffer pool   ┌ i.e. bring in many pages

Tuple identification
- fixed-len offset — index of each col match   i.e. each col store has same order
- embedded ids — ids attached to data   ↑ not common

Variable-length data   → store as dict with fixed-len keys elsewhere, then put keys in array

| Modern: cheap object store elsewhere

▷ PAX Storage Model
└ partion horizontally, then col-store per group



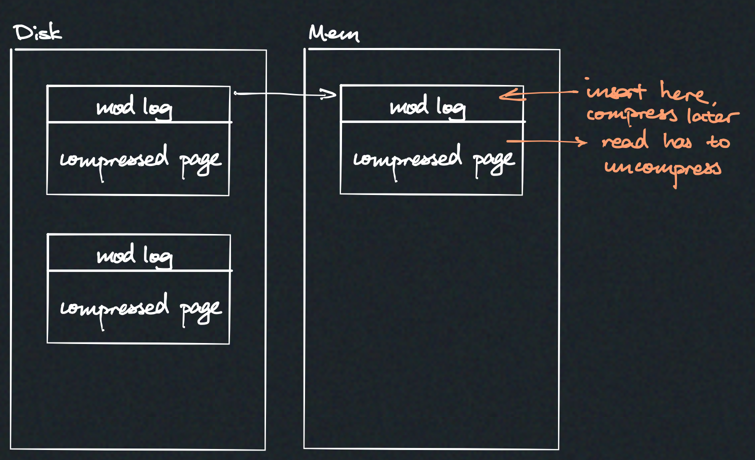
Compression — usually 2x perf boost

Goals
- produce fixed-len values
- when querying, decompress as late as possible
- lossless

Granularity
- record level
- attribute level
⋮

▷ Naïve compression — just use compression alg.
→ LZO, LZ4, ...

InnoDB compression



→ Can query compressed data by compressing query

Techniques
- Run-length encoding (col-major)
  (value, start idx, length)
  → can sort for better compression
- Bit packing
  → not use full range of i32
  → handle outliers different
- Patching
  put out of range data elsewhere, put marker for outlier
- Bitmap encoding aka one-hot encoding
  for cols without many values

  e.g. $$\begin{array}{c} Y \\ Y \\ N \end{array} \rightarrow \begin{array}{cc} Y & N \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{array}$$
- Delta encoding
  if consecutive changes small, just store the changes
- Dictionary Compression
  map distinct vals in col to short identifier
  reconstruct by dict lookup
  → sorting will still allow range query "order-preserving encoding"
  → Data struct: array, hashmap