Databases HW 3

Kevin Chen, EECS 339

1. File Organizations

- Search based on ranges: clustered/unclustered B+ Tree (clustered probably best, but you can only cluster once). Sorted is also okay. Basically, all of these provide a sorting mechanism such that we'd be able to binary search our way to the solution, instead of having to do a less efficient linear scan.
- Order doesn't matter: heap. Since order (or many other things) don't really matter, we might as well minimize overhead.
- Search based on field value: an unclustered hash index on that field would be much, much faster than everything else, since we won't have to go through all of the pages.

2. Disk

a) `Math.floor(1024 / 100) = 10`

b) Blocks: `100 000 / 10 = 10000`. Surfaces: `1` (`2000 * 50 / 2 = 50000` blocks per surface).

c) `50000 blocks * 2 sides * 5 platters * 10 records = 5 000 000` total records

d) 25 blocks per track, so the next surface would start at `26`. If the disk could read from all heads in parallel, then the next surface would be read after the first would be, so the page is `2`.

e) `400 * 1 / 5400 * 60 + 40 * 0.01 = 4.8s`. If parallel was possible, we'd get a speed up for 10x for transfer, so we'd get `0.84s`.

f) `100000 * (6ms + 10ms) + 400 * 60s / 5400 = 164.4s`

3. Buffer Pool

a) LRU: $5\ hits/10\ requests = 0.5$

b) Clock: 0.5

c) MRU: 0.4

Clearly, we want the highest hit rate, which is a tie between LRU and clock replacement. Since they're equal, clock replacement is probably better since it has lower overhead.

4. B+ Tree

a) identical: `25 26 13 15 20`, not identical: `1 4 6 8 9`

b) 3 entries (e.g. `1 4 6`)

c) 3 entries (doesn't make a difference. There's honestly just no reason it would make a difference at all, since duplicates wont cause maxed out buckets any quicker)

5. Parent Pointers

a) The children parent pointers aren't valid anymore -- none of them will be pointing at N2, but all at N.

b) One solution could be to immediately recurse back down on any insertion to update the child parent pointers. A second solution is to do this all in bulk periodically or at some point when the page is needed.

c) Basically, both of these solutions could lead to pretty significantly slow downs from more work on each op (first) or unnecessary read writes (second), while contributing not really anything of value.

d) Parent pointers don't contribute anything positive while slowing things down.