Indexing

Workflow Recap:

- **Load/Parse data records**
- **Filter records (rank ≠ Ensign, age > 25)**
- **Print names**

Today: Let's combine steps 1 & 2

Basics: Sorting

Root Idea: Sort records by age

Algorithm:

- Can use Binary Search over data to find first record > 25
- Return that record and everything following it

Challenges

Need a record format with predictable record locations

- Fixed-Size Records
- Put a fixed number of records on each "page"

- Paging makes binary search pricy
- What happens if the data changes?

Does this generalize?

- age > X; Same as above
- age < Y; Yes, Start from first record, return everything until first record >= X
- age = X: Yes, Binary Search Still Works (may still need to return multiple records)
- X < age < Y; Yes, Binary Search, then return everything until first record >= Y

Indexes

Challenges

Paging (respectively cache lines) makes binary search expensive

- Scan is still comparatively cheap

What if we need to access 2 (or more) attributes?

- Modulo a few corner cases, we can't sort more than once
- No real answer for this point today… we'll get back to it

Idea 1: Page-aware 'Key' Summaries

Implementation 1: One page of summaries

- Fit as many [key+pointer] pairs as you can in one page
- Each pointer points to the first record equal to or greater than the listed key
- Binary search on keys to find the pointer to follow

- **Limitation**: Doesn't scale to larger data sizes; Still may need to binary search across data on multiple pages)

- **Implementation 2: Add indirection (Tree Indexes)**
  - Binary search within a page is cheap, so keep one [key+pointer] per page
  - Pack as many [key+pointer]s into a summary page as you can.
  - If you overflow the summary page, start building a summary of summaries
    - Tier 1: Data Pages
    - Tier 2: Pages of [Key+Pointer]s to the first key on each data page
    - Tier 3: Pages of [Key+Pointer]s to the first key on each tier 2 page
    - Tier 4: etc…
- **Challenge: Handling Changing Data**
  - Can't insert into the middle of a sorted file
  - Can't insert into a packed (sorted) summary page
- **Implementation 3: Out-of-order pages (B+Tree-Ish Indexes)**
  - Treat pages as atomic blobs of storage (rather than a single contiguous region)
    - **Bonus**: Don't need fixed-size records
  - Leave empty space on each data page and each summary (tree) page
  - What to do when a page "fills up" or "empties out"?
    - Shift records to/from other pages at the same level (pivot)
    - Merge two pages together
    - Create a new level / flatten a level
  - Degenerate case:
    - Super-tall structure
- **Implementation 4: As above, but maintain size invariant (B+Tree)**
  - **Invariant 1**: Uniform Tree Depth
  - **Invariant 2**: 50% ≤ fill ≤ 100% (for all except root page)
  - When page drops below 50% fill, merge with adjacent page
    - Recur higher if necessary
  - When page exceeds 100% fill, split into 2 pages
    - Recur higher if necessary
  - When root drops to 1 pointer, reduce depth by 1
  - When root exceeds capacity, increase depth by 1
  - **Optimization**: Borrow/Loan records/[key+pointer]s from/to adjacent pages