Recap

Supporting Multiple Attributes

- **Idea 1**: Build separate "clustered" indexes for each attribute of interest
  - **Pro**: Super fast for reads
  - **Con**: Lots of space, slow to update

- **Idea 2**: Hierarchical indexes - Organize according to 2+ attributes
  - **Pro**: Super space-efficient
  - **Con**: Doesn't support every type of query
    - Given an index with attributes $A_1, A_2, \ldots, A_N$:
      - Can (easily) support any query of the form ($C_i$ are constants): $A_1 = C_1$ AND $A_2 = C_2$ AND \ldots AND $A_K < C_K$ (for any $K \leq N$)
        - $A_K$ can have any range predicate on it ($<$, $>$, $\leq$, $\geq$, BETWEEN, ...)
        - $A_1$ to $A_{K-1}$ can only have equality predicates
    - **Adjustment**: R-Like Trees (maybe will discuss later on in the term)

- **Idea 3**: Build a "secondary" index for each attribute of interest
  - **Pro**: Not as much space (particularly for large records), faster updates
  - **Con**: Slower (need 2 rounds of access per record... potentially out of order)
    - **Adjustment**: Load all keys into memory from the second index, sort, then, "scan" over primary index
      - **Limitation**: Need enough memory to keep the keys in memory

Supporting Updates

- **Idea 1**: Create a separate "Holding Area" for new records
  - Index/sort holding area separately, periodically merge with overall dataset.
    - **Limitation**: Lots and lots of copies per record (data "locked" while updating)

B+Trees

- **Idea 3**: Leave some "wiggle room" in pages.
  - **Ideas**:
    - Allow data (and index) pages to not be full
    - Drop the requirement that data be in a contiguous region
  - **Questions**
    - How much space to reserve?
      - Too much space reserved: Structure ends up being too tall
      - Too little space reserved... then what?
    - What to do when a page "fills up" or "empties out"?
      - Borrow/Lend records to/from other pages at the same level
      - Merge two pages together
- Create a new level / flatten a level

  ▼ Observation: Lower bound of 50% fill = Max 2x Depth
  - (error in previous notes... depth could still double)
  - 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

  ▼ What if we can’t merge with adjacent records?
  - **Adjustment:** Borrow/Loan records/[key+pointer]s from/to adjacent pages

  ▼ Worst case behavior
  - ▼ Alternating Insertions / Deletions occurring on a 50%/100% boundary:
    - Every insert triggers a split
    - Every delete triggers a merge
    - Doesn’t happen very often...
    - Borrow/Loan help prevent this
  - Other ideas: Background task to continuously rebalance tree away from dangerous split/merge thresholds