07 – Tree Indexes 1

1. Table index
    A. Synchronized replica of a subset of a table's attributes that are organized for access using a subset of attributes
    B. Storage & Maintenance Overhead
2. B+Tree
    A. Self balancing tree structure
    B. Keeps data sorted
    C. O(logN) complexity
    D. B+Tree has records only in leaf nodes
    E. Advantage
        i. Clustered Index
            1. Records are sorted in order of primary key (hash table is not)
        ii. Selection Condition
            1. If query has data for some attributes, Tree structure can search that data (hash table need all of attributes)
3. Design Decisions
    A. Node Size : Disk slow? Enlarge node size!
    B. Delayed Merge : Merge may occur more re-organization
    C. Variable length keys
        i. Pointers
        ii. Various size of nodes
        iii. Padding
        iv. Key mapping
    D. Non-unique keys : Duplicate or List?
    E. Intra-Node Search : Linear or Binary (or Interpolate)

4. Optimization
    A. Prefix Compression - common prefix only stored once
    B. Suffix Truncation - keys are only used as traffic, So, only needed part stored
    C. Bulk Insert - sort data first, construct tree by bottom up logic
    D. Pointer Swizzling –
       if page is pinned, don’t lookup the page table, just get poiner
5. What is it?
    A. Pointer Swizzling (I didn’t get it well)
6. Introduced Papers
    A. Efficient Locking for Concurrent Operations on B-Trees
    B. Modern B-Tree Techniques