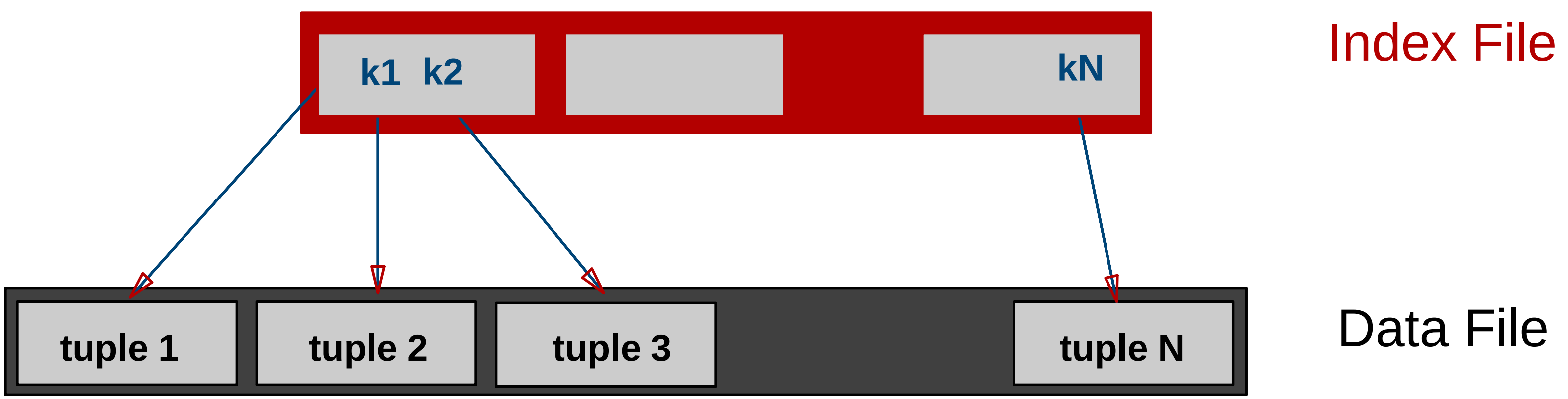
Indexing

Ramakrishnan/Gehrke Ch. 8

"How index-learning turns no student pale
Yet holds the eel of science by the tail."
-- Alexander Pope (1688-1744)

Range Searches

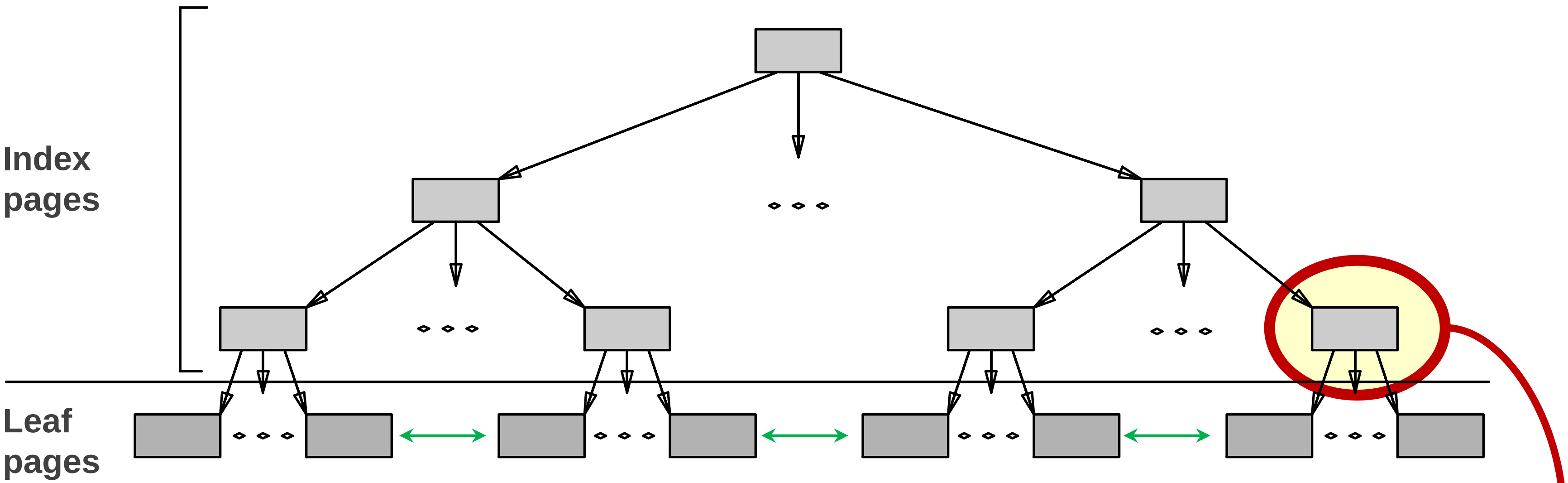
- *``Find all students with gpa > 3.0''*
  - sorted file (by gpa!), fixed-length records:
    binary search to find first student, then scan to find rest
  - Cost of binary search can be quite high
- Simple idea:
  Create an `index' file containing only key values + search values
  - Can do binary search on (smaller) index file!



Indexes

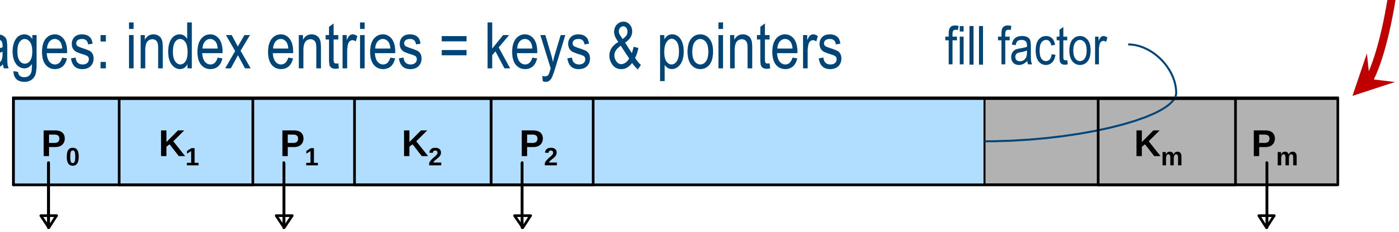
- speeds up selections on predefined search key field(s)
  - one relation (~file)
  - Any attribute (except BLOB) can be search key for an index on the relation
- collection of data entries
  - For efficient retrieval of all data entries k* for given key value k
- Index vs sorted files
  - Both: search faster than just heap
  - Updates: index much faster

B+ Tree Indexes



- Ordered, balanced tree of degree m

- Non-leaf pages: index entries = keys & pointers

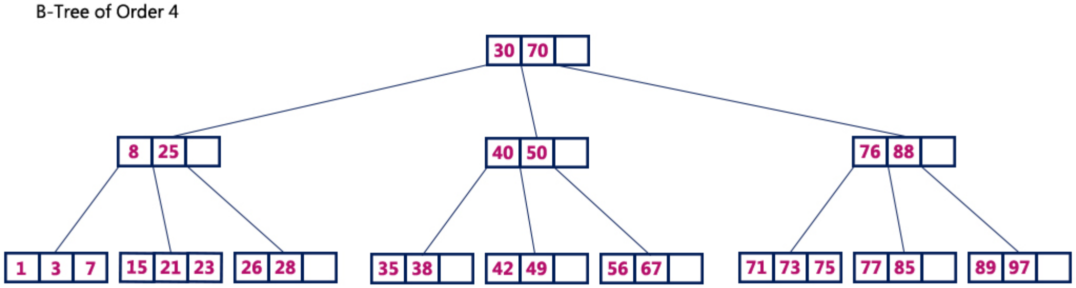


- Leaf pages: keys + data pointers; prev/next page chain

B+-Tree Definition

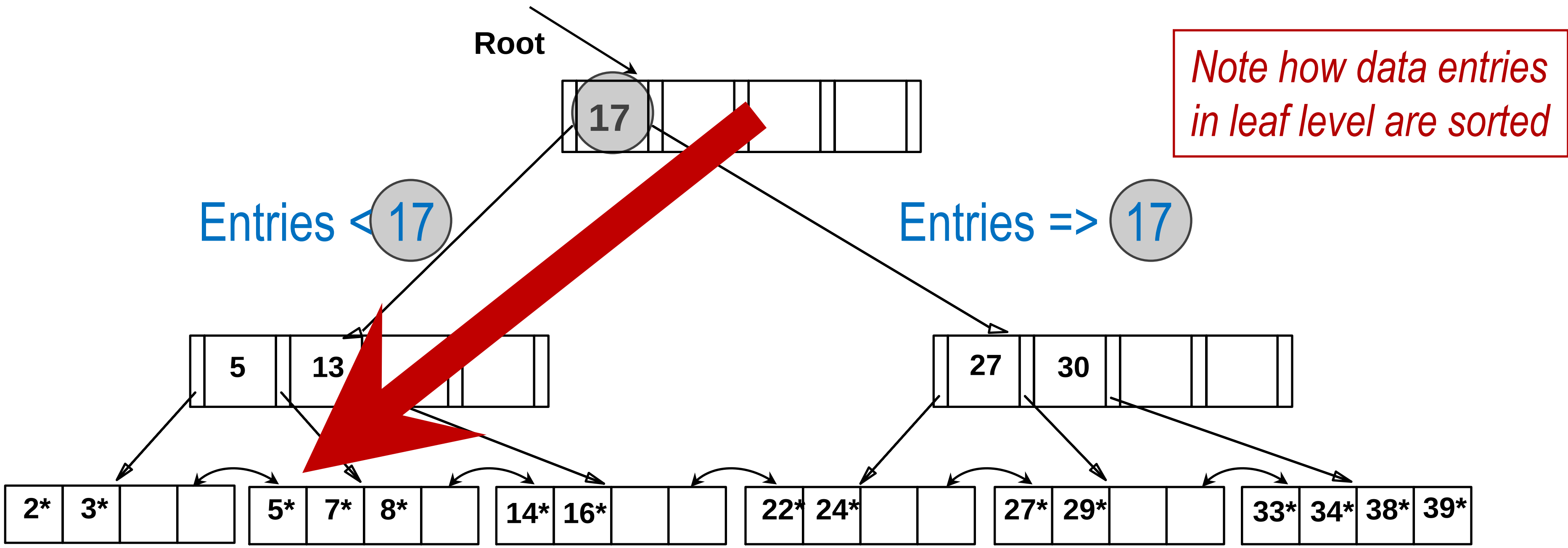
- B+-Tree of Order m has the following properties...
  - #1 - All leaf nodes at same level.
  - #2 - nodes except root have [m/2]-1 … m-1 keys.
  - #3 - non leaf nodes except root (i.e. all internal nodes) have at least m/2 children.
  - #5 - non leaf node with n-1 keys have n number of children.
  - #6 - key values in a node sorted in ascending order.

https://www.cs.usfca.edu/~galles/visualization/BTree.html



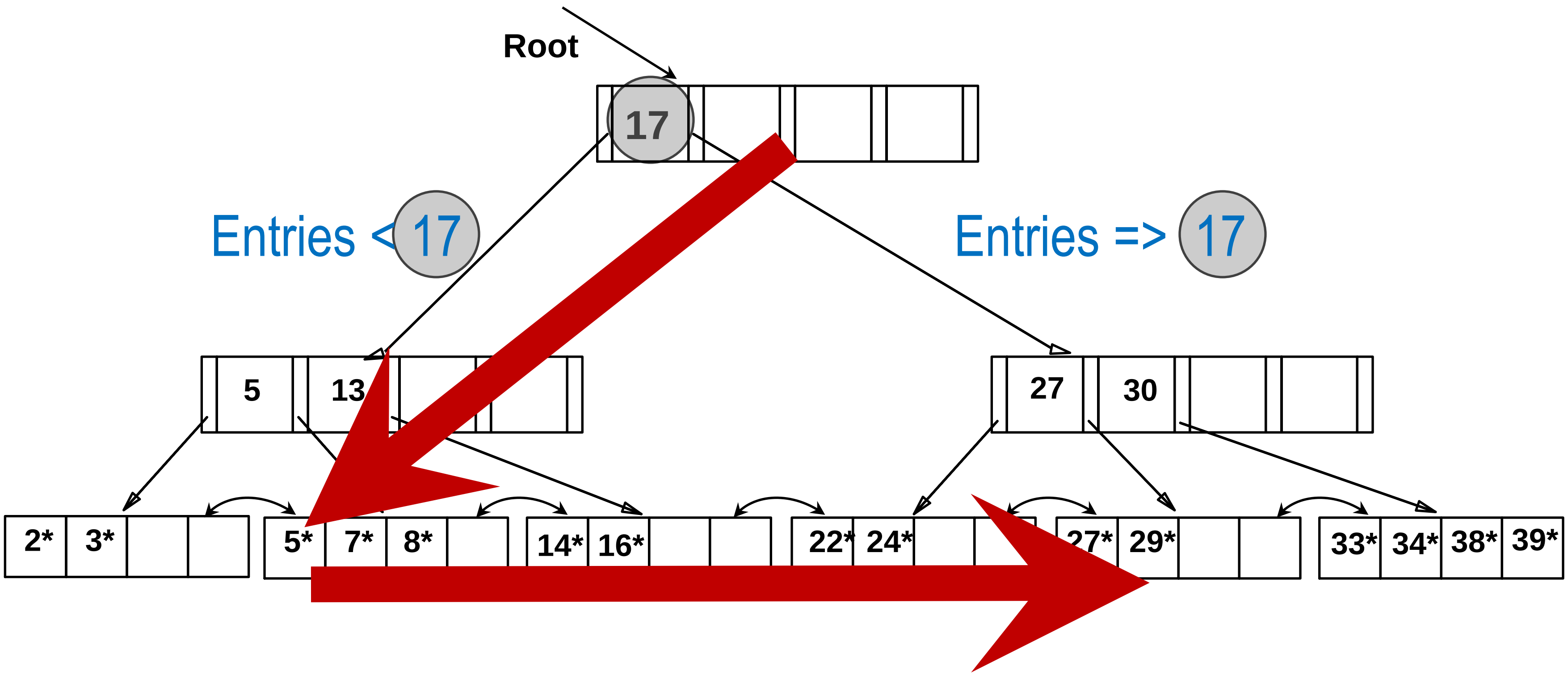
[http://btechsmartclass.com/data_structures/b-trees.html]

B+ Tree: Point Search



- Find 28*? 29*?

B+ Tree: Range Search

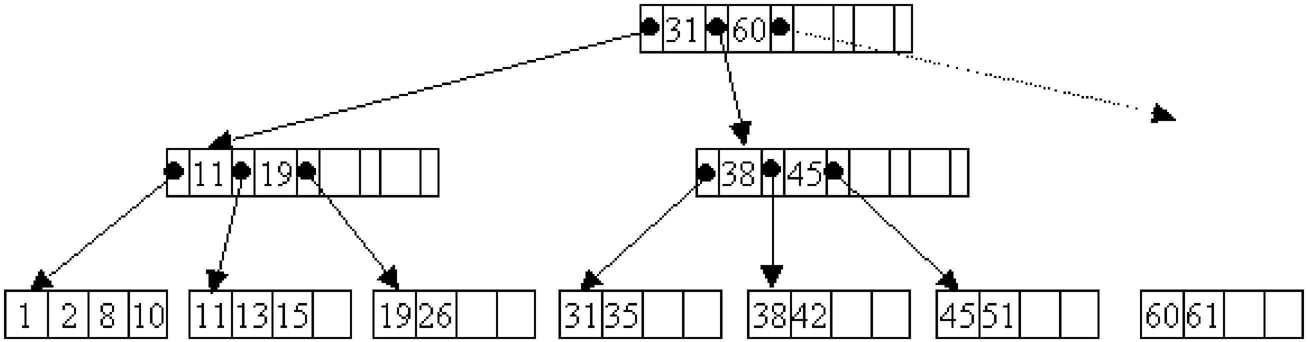


- all between 15* and 30*?

Note how data entries in leaf level are sorted

More Exercises

- Consider
  - # node reads from disk – determines speed
  - # comparisons – not performance relevant, but for understanding mechanics



- Find 15, 20, 0
- Find all 11 – 15; 20 – 32

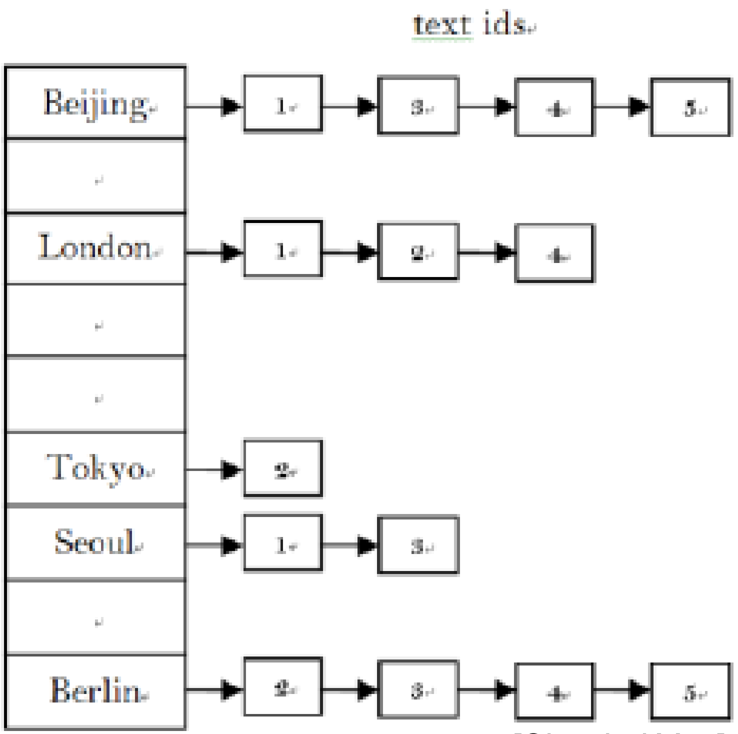
[https://condor.depaul.edu/ntomuro/courses/417/notes/lecture3.html]

B+ Trees in Practice

- Typical fill-factor: 67% (outdated; today ~90%)
- Average fan-out: 133
- Typical capacities:
  - Height 3: 133^3 = 2,352,637 records
  - Height 4: 133^4 = 312,900,700 records
- Can often hold top levels in buffer pool:
  - Level 1 = 1 page = 8 Kbytes
  - Level 2 = 133 pages = 1 Mbyte
  - Level 3 = 17,689 pages = 133 MBytes

Hash-Based Indexes

- Goal: *compute* address without disk access
  - get data in O(1)
- Idea: distribute data evenly into fixed number of "buckets"
  - Compute location from key via Hashing function
  - Ex: h(int r) = r**a* mod *b*, *b* prime relative to *a*
  - overflow pages
- Hash index = bucket set + hashing function
  - Bucket = primary page + 0..n overflow pages
- only equality, no range queries



[Shankai Yan]

Index-Only Plans

- Index can answer queries without retrieving tuples from relations
  - Simple index: *<E.dno>*

    ```
    SELECT E.dno, COUNT(*)
    FROM Emp E
    GROUP BY E.dno
    ```

  - Composite index: *<E.dno,E.sal>*

    ```
    SELECT E.dno, MIN(E.sal)
    FROM Emp E
    GROUP BY E.dno
    ```

  - More complex example: *<E. age,E.sal>* or *<E.sal, E.age>*

    ```
    SELECT AVG(E.sal)
    FROM Emp E
    WHERE E.age=25 AND
     E.sal BETWEEN 3000 AND 5000
    ```

Index Selection Guidelines

- For each query in workload:
  - relations accessed? attributes retrieved? selection/join conditions? How selective?
- For each update in workload:
  - Type of update (INSERT/DELETE/UPDATE) + attributes affected
  - attributes involved in selection/join conditions? How selective?
- Trade-off: Indexes can make queries faster, updates slower
  - …and require disk space
- …a practitioner's approach:
  - Consider most important queries in turn, improve only where necessary

Summary

- Index = “summary file” to quickly find tuples
  - Can have several indexes on table
  - Hash-based for equality search
  - Tree-based for range search, equality search
- Essential for tuning:
  - Understanding query workload
  - clear performance goals