Tree-Structured Indexes

CMPSCI 445
Fall 2008
Review

- As for any index, 3 alternatives for data entries $k^*$:
  - Data record with key value $k$
  - $\langle k, \text{rid of data record with search key value } k \rangle$
  - $\langle k, \text{list of rids of data records with search key } k \rangle$
- Choice is orthogonal to the indexing technique used to locate data entries $k^*$.
- Tree-structured indexing techniques support both range searches and equality searches.
**B+ Tree: Most Widely Used Index**

- Inserts/deletes keep tree *height-balanced*. Log \( F \) \( N \) cost (\( F \) = fanout, \( N \) = # leaf pages).
- Minimum 50% occupancy (except for root). Each node contains \( d \leq m \leq 2d \) entries, where \( d \) is called the *order* of the tree.
- Supports equality, range-searches, updates efficiently.

![Diagram of B+ Tree](image-url)
Example B+ Tree

- Search begins at root, and key comparisons direct it to a leaf.
- Search for 5*, 15*, all data entries >= 24* ...

Based on the search for 15*, we know it is not in the tree!
B+ Trees in Practice

- Typical order: 100. Typical fill-factor: 67%.
  - average fanout = 133

- Typical capacities:
  - Height 4: $133^4 = 312,900,700$ records
  - Height 3: $133^3 = 2,352,637$ 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
Inserting a Data Entry into a B+ Tree

- Find correct leaf \( L \).
- Put data entry onto \( L \).
  - If \( L \) has enough space, done!
  - Else, must split \( L \) (into \( L \) and a new node \( L_2 \))
    - Redistribute entries evenly, copy up middle key.
    - Insert index entry pointing to \( L_2 \) into parent of \( L \).

- This can happen recursively
  - To split index node, redistribute entries evenly, but push up middle key. (Contrast with leaf splits.)

- Splits “grow” tree; root split increases height.
  - Tree growth: gets wider or one level taller at top.
Previous Example

Inserting 8*
Inserting 8* into Example B+ Tree

- Observe how minimum occupancy is guaranteed in both leaf and index page splits.
- Note difference between copy-up and push-up; be sure you understand the reasons for this.

Entry to be inserted in parent node. (Note that 5 is copied up and continues to appear in the leaf.)

Entry to be inserted in parent node. (Note that 17 is pushed up and only appears once in the index. Contrast this with a leaf split.)
Example B+ Tree After Inserting 8*

- Notice that root was split, leading to increase in height.
- In this example, we can avoid split by re-distributing entries between siblings; but not usually done in practice.