Tree primary organizations

• Tree terminology:
  – Order: max number of children per node
  – Level of a node: number of nodes in the path from the root to the node
  – Height of a tree: Maximum level of a node
  – Balanced tree: levels of leaf nodes differ by at most 1
Tree primary organizations

- Binary tree
- B-tree
B-tree

- A B-tree is a perfectly balanced search tree in which nodes have a variable number of children.
- Here, let ‘k*’ denote the full record with key k, and a tree node be a page.
B-tree

• A B–tree of order $m$ ($m \geq 3$) is perfectly balanced and has the following properties:
  – Each node has at most $(m - 1)$ keys and, except the root, at least $\lceil m/2 \rceil - 1$ keys
  – A node with $j$ keys has also $p_0,\ldots,p_j$ ($j + 1$) pointers to distinct subtrees, undefined in the leaves. Let $K(p_i)$ be the set of keys in the subtree $p_i$
  – Each non leaf node has the following structure
B-tree

\[ [p_0, k_1^*, p_1, k_2^*, p_2 \ldots k_j^*, p_j] \]

\[
\begin{align*}
K(p_0) & \quad \forall y \in K(p_0) \quad y < k_1 \\
K(p_1) & \quad \forall y \in K(p_1) \quad k_1 < y < k_2 \\
K(p_j) & \quad \forall y \in K(p_j) \quad y > k_j
\end{align*}
\]
B-tree

Equality search: k = 5

Range search: k >= 23
B-tree

• Relationship between the height $h$, the order $m$ and number of keys $N$:

• Example: record 100 byte, pointer 4 byte, a page 4096 byte, $m = 40$ ((4096-4)/(100+4) + 1)
  
  – $h=1$ Nodes = 1 $\quad$ NMax = 39

  – $h=2$ Nodes = 1+40 $\quad$ NMax = (1+40)*39 = 1599

  – $h=3$ Node = 1+40+1600 = 1641

  $\quad$ NMax = 1641*39= 63999

• $\log_m(N + 1) \leq h \leq 1 + \log_{[m/2]}\left(\frac{N+1}{2}\right)$
B-tree: search cost

• Equality search \((k = v)\): \(1 \leq C \leq h\)

• Range search \((p = (v_1 \leq k \leq v_2))\):
  
  \(- s_f(p) = (v_2 - v_1)/(k_{\text{max}} - k_{\text{min}})\)
  
  \(- E_{\text{reg}} = s_f(p) \times N\)
  
  \(- C = s_f(p) \times N_{\text{nodes}}\)
  
  \(- h \leq C \leq N_{\text{nodes}}\)
Insertion

• Insertion in an unfull leaf
• Insertion in a full leaf ...
Insertion of 6

5 moves in the ancestor node....
The tree height increases

In the worst case, the insertion cost is $h$ reads + $(2h+1)$ writes
Deletion

• The key is in a nonleaf node: it is replaced by the next key, which is in a leaf node, and is deleted from there
• The key is in a leaf node: it is deleted
• What happens if, after deletion, the leaf node has less than \(\lceil m/2 \rceil - 1\) elements?
Rotation

Deletion of 16 and rotation
Merging

Deletion of 22, 20 and **merging**
Deletion: cost

• In the worst case (merging at all levels and rotation at the root children), the cost is:
  - \((2h - 1)\) reads + \((h+1)\) writes
Note: when a leaf splits, a copy of the key is inserted the ancestor (B⁺-tree), when a nonleaf node splits, a key moves in the ancestor (B-tree)
B+-Tree: Equality Search Cost

Equality search \((k = v_1)\)

\[ C = 1 \quad (C = 2 \quad \text{or} \quad C = 3) \]

Range search \((p = (v_1 \leq k \leq v_2))\)

\[ s_f(p) = (v_2 - v_1)/(k_{max} - k_{min}) \]

\[ C = s_f(p) \cdot N_{leaf} \]

Let us consider the leaf access cost only
Deletion

• Search the leaf F with the key
• Actual deletion:
  – If F does not underflow, end
  – Otherwise, apply merging or rotation
  – If a merging is performed, delete a key from the ancestor of F, in the B-tree structure...
Secondary organizations: indexes

• An index is a mapping of attribute(s) (key) values to RID of records.

• Definition. An index I on an attribute (key) K of a relational table R is an ordered table I(K, RID)

• A tuple of the index is a pair (k_i, r_i), where k_i is a key value for a record, and r_i is a reference (RID) to the corresponding record.

• We can have several indexes on a table, each with a different search key
## Examples

### Table

<table>
<thead>
<tr>
<th>RID</th>
<th>StudCode</th>
<th>City</th>
<th>BirthYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>MI</td>
<td>1972</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>PI</td>
<td>1970</td>
</tr>
<tr>
<td>3</td>
<td>102</td>
<td>PI</td>
<td>1971</td>
</tr>
<tr>
<td>4</td>
<td>104</td>
<td>FI</td>
<td>1970</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>MI</td>
<td>1970</td>
</tr>
<tr>
<td>6</td>
<td>107</td>
<td>PI</td>
<td>1972</td>
</tr>
</tbody>
</table>

### Index on StudCode

<table>
<thead>
<tr>
<th>StudCode</th>
<th>RID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>2</td>
</tr>
<tr>
<td>102</td>
<td>3</td>
</tr>
<tr>
<td>104</td>
<td>4</td>
</tr>
<tr>
<td>106</td>
<td>5</td>
</tr>
<tr>
<td>107</td>
<td>6</td>
</tr>
</tbody>
</table>

### Index on BirthYear

<table>
<thead>
<tr>
<th>BirthYear</th>
<th>RID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>2</td>
</tr>
<tr>
<td>1970</td>
<td>4</td>
</tr>
<tr>
<td>1970</td>
<td>5</td>
</tr>
<tr>
<td>1971</td>
<td>3</td>
</tr>
<tr>
<td>1972</td>
<td>1</td>
</tr>
<tr>
<td>1972</td>
<td>6</td>
</tr>
</tbody>
</table>
Clustered Indexes

• Clustered vs. unclustered
• If the order of data records is the same as the order of data entries, then it is called clustered index.
• Clustered = with data almost ordered, if there are insertions
Data organizations for two keys: $K_p$ and $K$
Clustered vs. Unclustered

Search cost

Equality search cost \((k = v_1)\)

Range search cost \((p = (v_1 \leq k \leq v_2))\)

\[ C_{\text{clustered}} = s_f(p) \times N_{\text{leaf}} + s_f(p) \times N_{\text{pag}} \]

\[ C_{\text{unclustered}} = s_f(p) \times N_{\text{leaf}} + s_f(p) \times N_{\text{rec}} \]
Summary

• A B-tree is a fully balanced dynamic structure that automatically adapts to inserts and deletes
• A B+-tree refine the B-tree to improve range search and sorted data scans
• Indexes are used for secondary organizations
• Types of Indexes: clustered vs unclustered