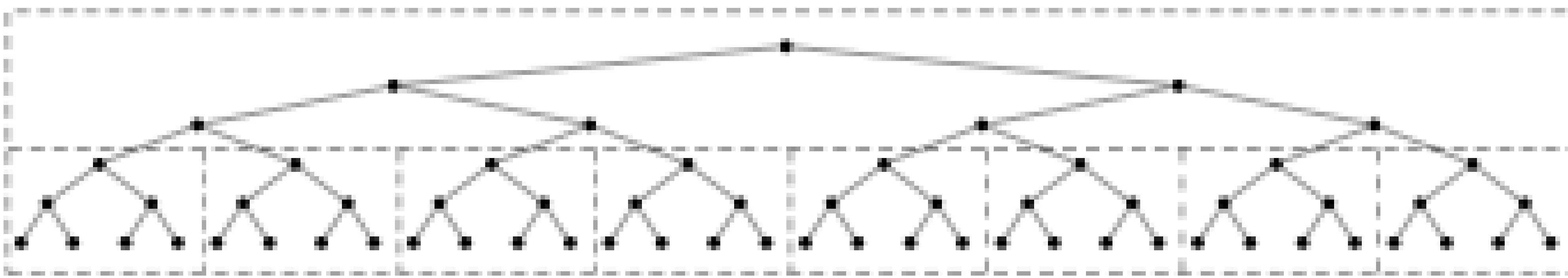
Tree primary organizations

- 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

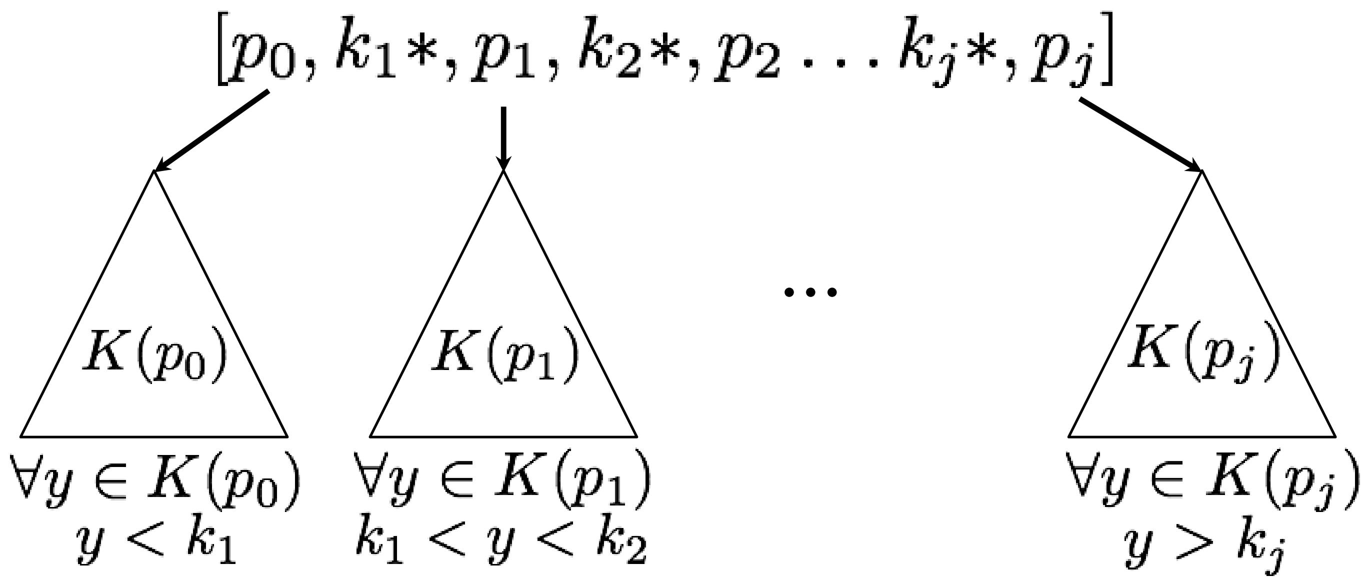
- A B-tree is a perfectly balanced search tree in which nodes have a variable number of children
- Here, let 'k*' denote the full record with key k, and a tree node be a page.

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 p0,…,pj (j + 1) pointers to distinct subtrees, undefined in the leaves. Let K(pi) be the set of keys in the subtree pi
  - Each non leaf node has the following structure

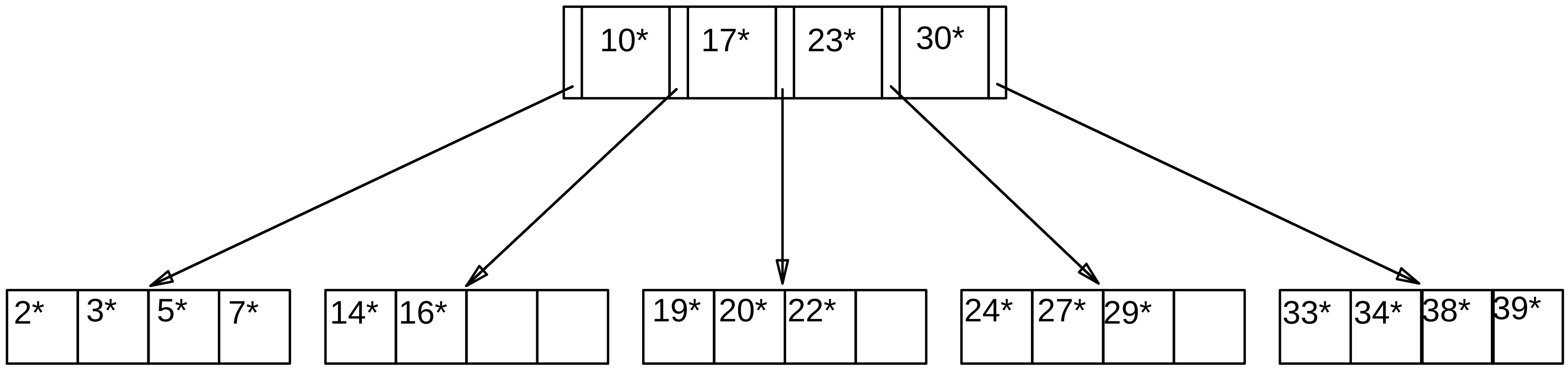
B-tree

- .



$[p_0, k_1*, p_1, k_2*, p_2 \ldots k_j*, p_j]$
$K(p_0)$
$K(p_1)$
...
$K(p_j)$
$\forall y \in K(p_0)$
$y < k_1$
$\forall y \in K(p_1)$
$k_1 < y < k_2$
$\forall y \in K(p_j)$
$y > k_j$

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 NMax = 39
  - h=2 Nodes = 1+40 NMax = (1+40)*39 = 1599
  - h=3 Node = 1+40+1600 = 1641
    NMax = 1641*39= 63999
- $\log_m(N+1) \le h \le 1 + \log_{\lceil m/2 \rceil}(\frac{N+1}{2})$

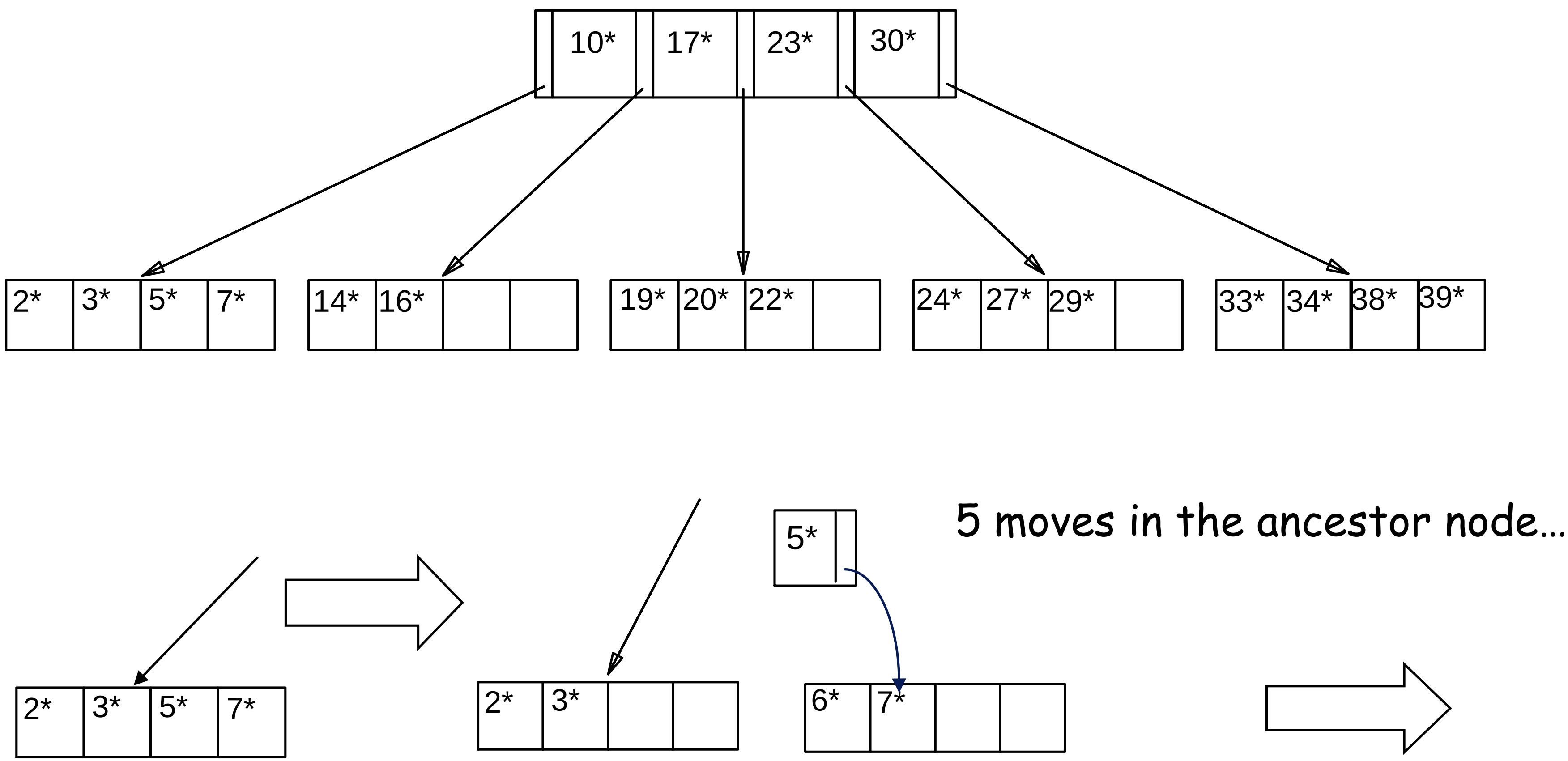
B-tree: search cost

- Equality search ($k = v$): $1 \le C \le h$
- Range search ($p = (v_1 \le k \le v_2)$):
  - $s_f(p) = (v_2 - v_1)/(k_{max} - k_{min})$
  - $E_{reg} = s_f(p) \times N$
  - $C = s_f(p) \times N_{nodes}$
  - $h \le C \le N_{nodes}$

Insertion

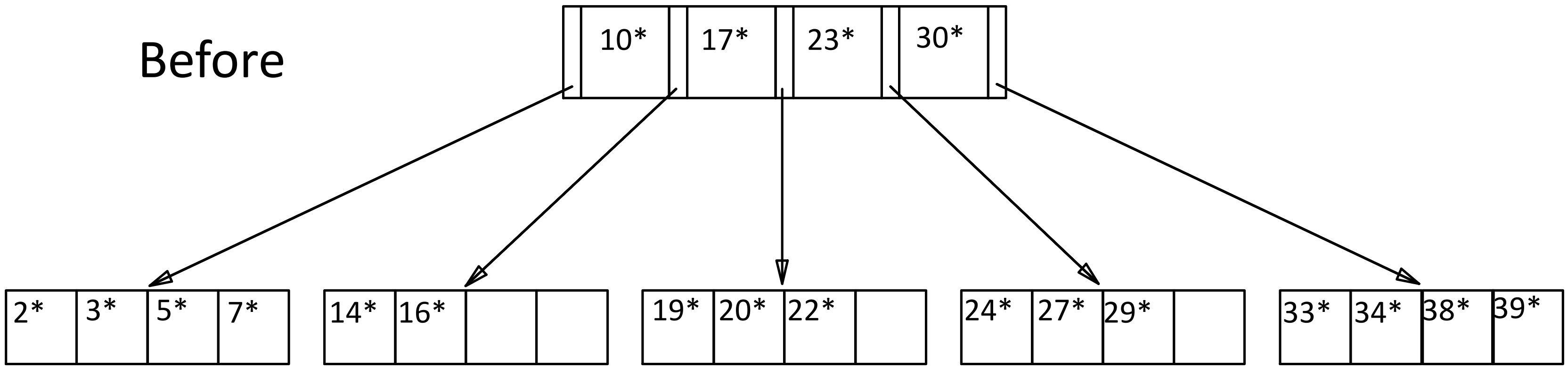
- Insertion in an unfull leaf
- Insertion in a full leaf ...

Insertion of 6



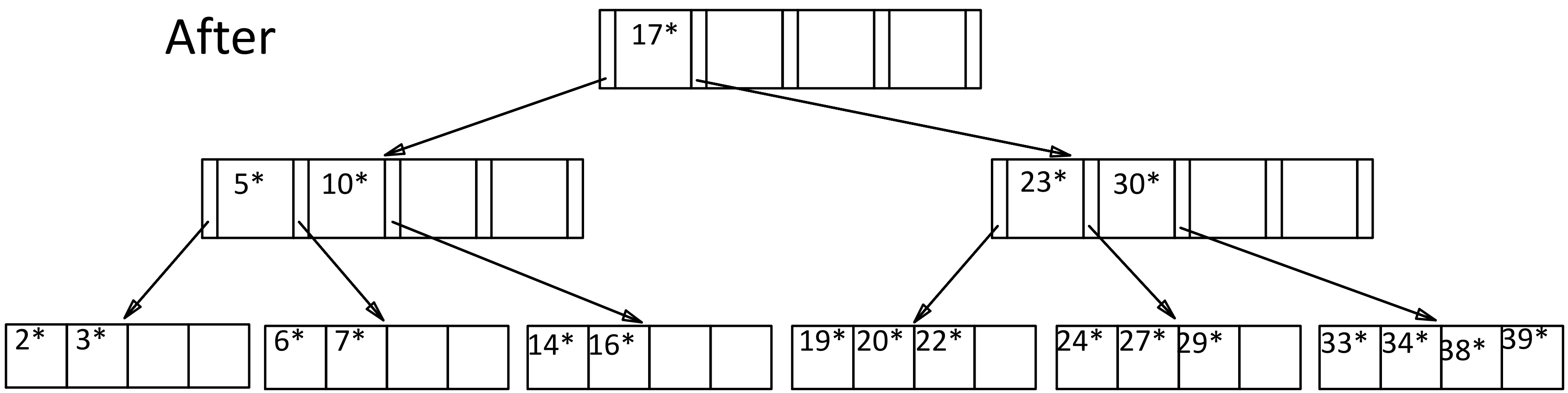
Insertion of 6

Before



The tree height increases

After

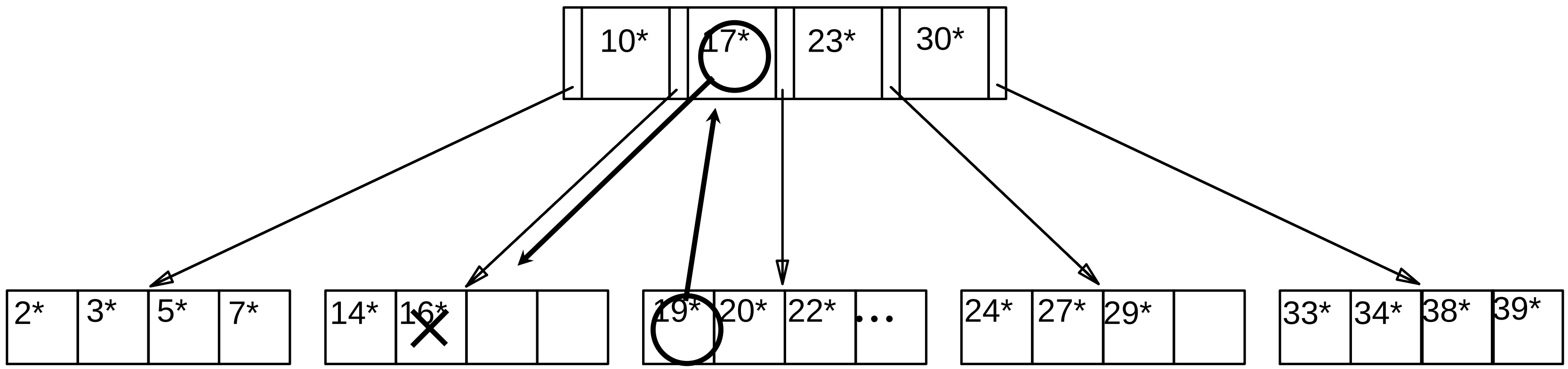


In the worst case, the insertion cost is h reads + (2h+1) writes

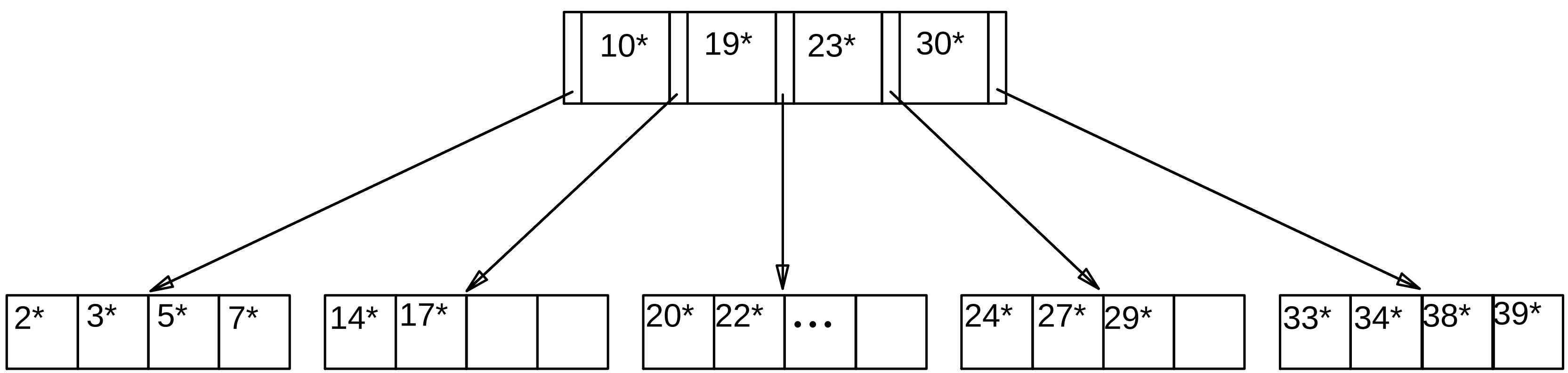
Deletion

- 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 ?

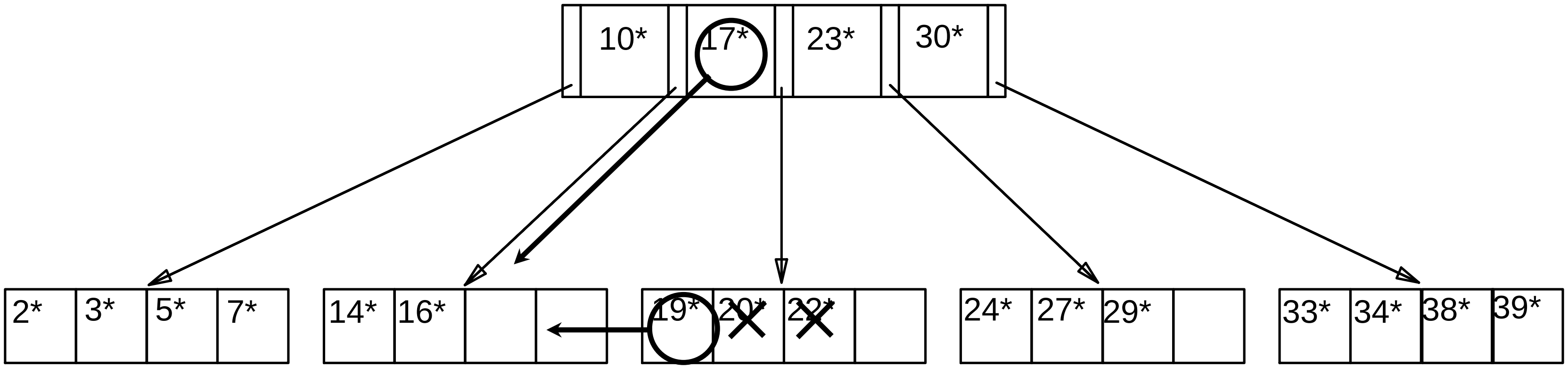
Rotation



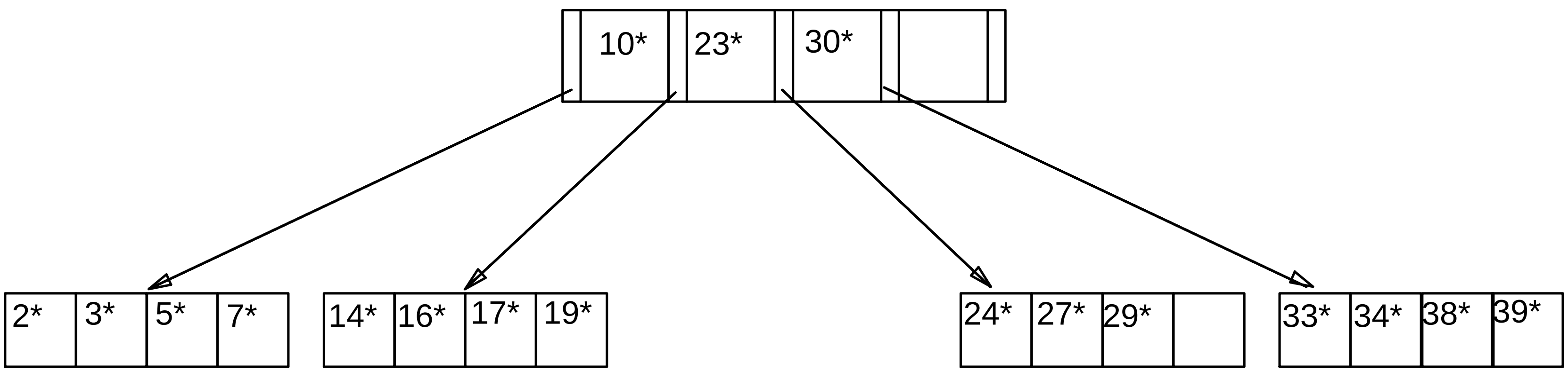
Deletion of 16 and **rotation**



Merging



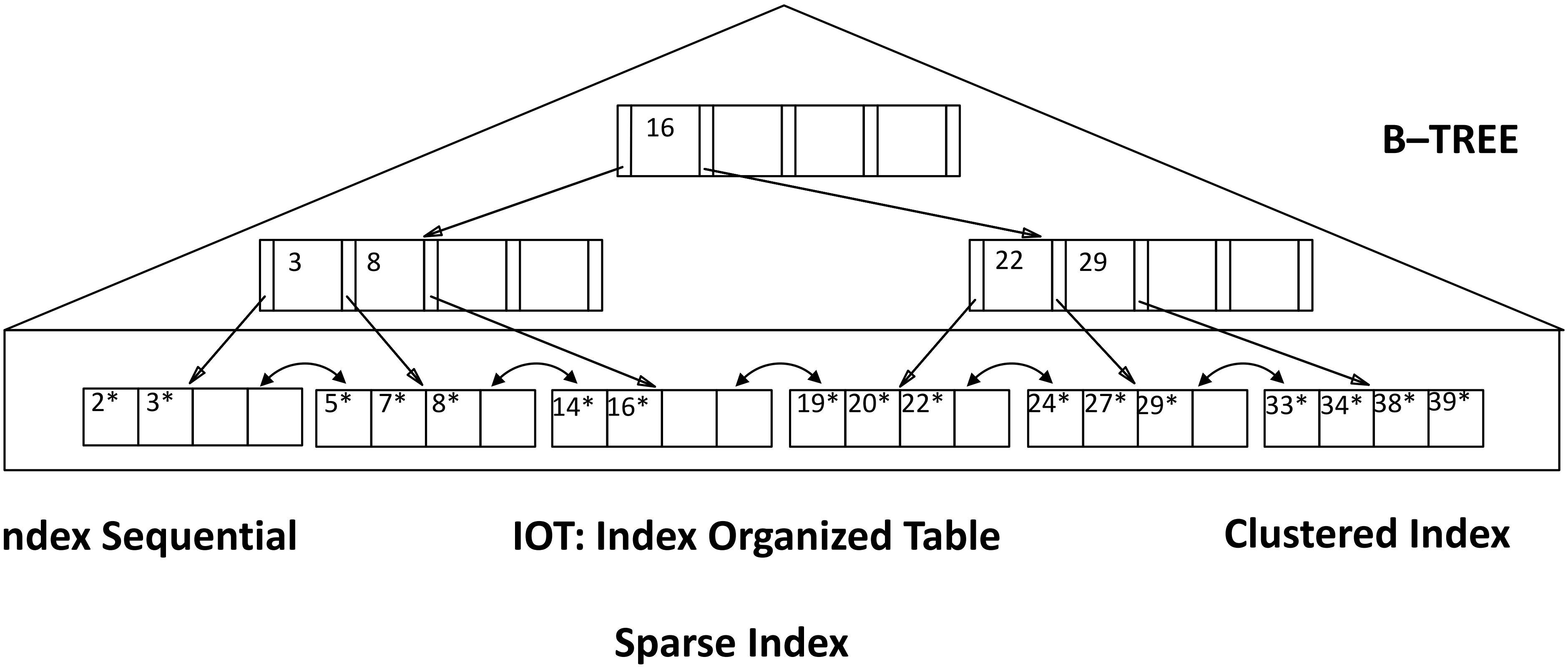
Deletion of 22, 20 and **merging**



Deletion: cost

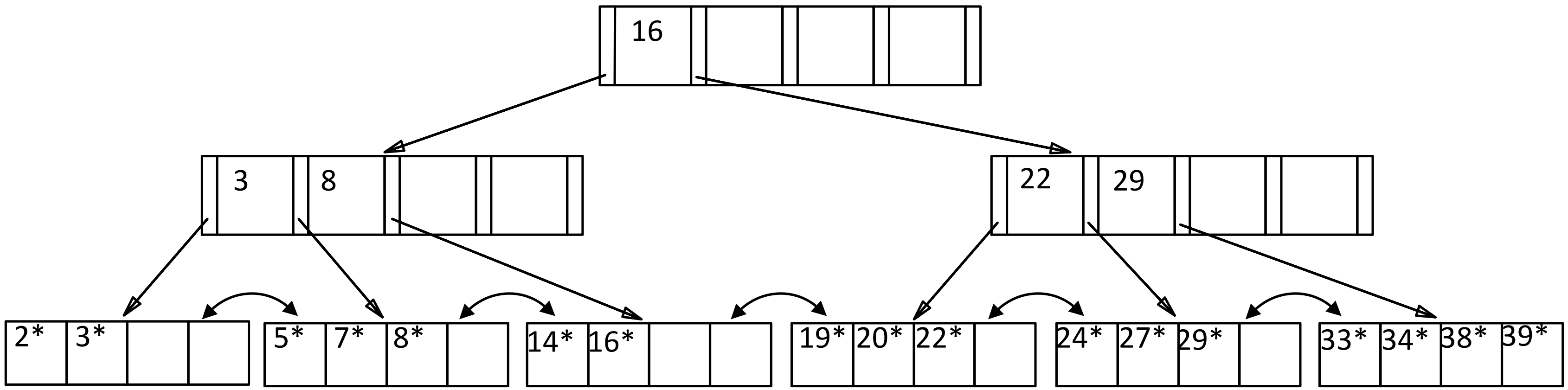
- In the worst case (merging at all levels and rotation at the root children), the cost is:
  - (2h – 1) reads + (h+1) writes

B^+-Tree



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



Let us consider the leaf access cost only

equality search ($k = v1$) $C = 1$ ($C = 2$ or $C = 3$)

range search ($p = (v1 \le k \le v2)$) $s_f(p) = (v2 - v1)/(k_{max} - k_{min})$

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

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

RID	StudCode	City	BirthYear
1	100	MI	1972
2	101	PI	1970
3	102	PI	1971
4	104	FI	1970
5	106	MI	1970
6	107	PI	1972

Index on StudCode

StudCode	RID
100	1
101	2
102	3
104	4
106	5
107	6

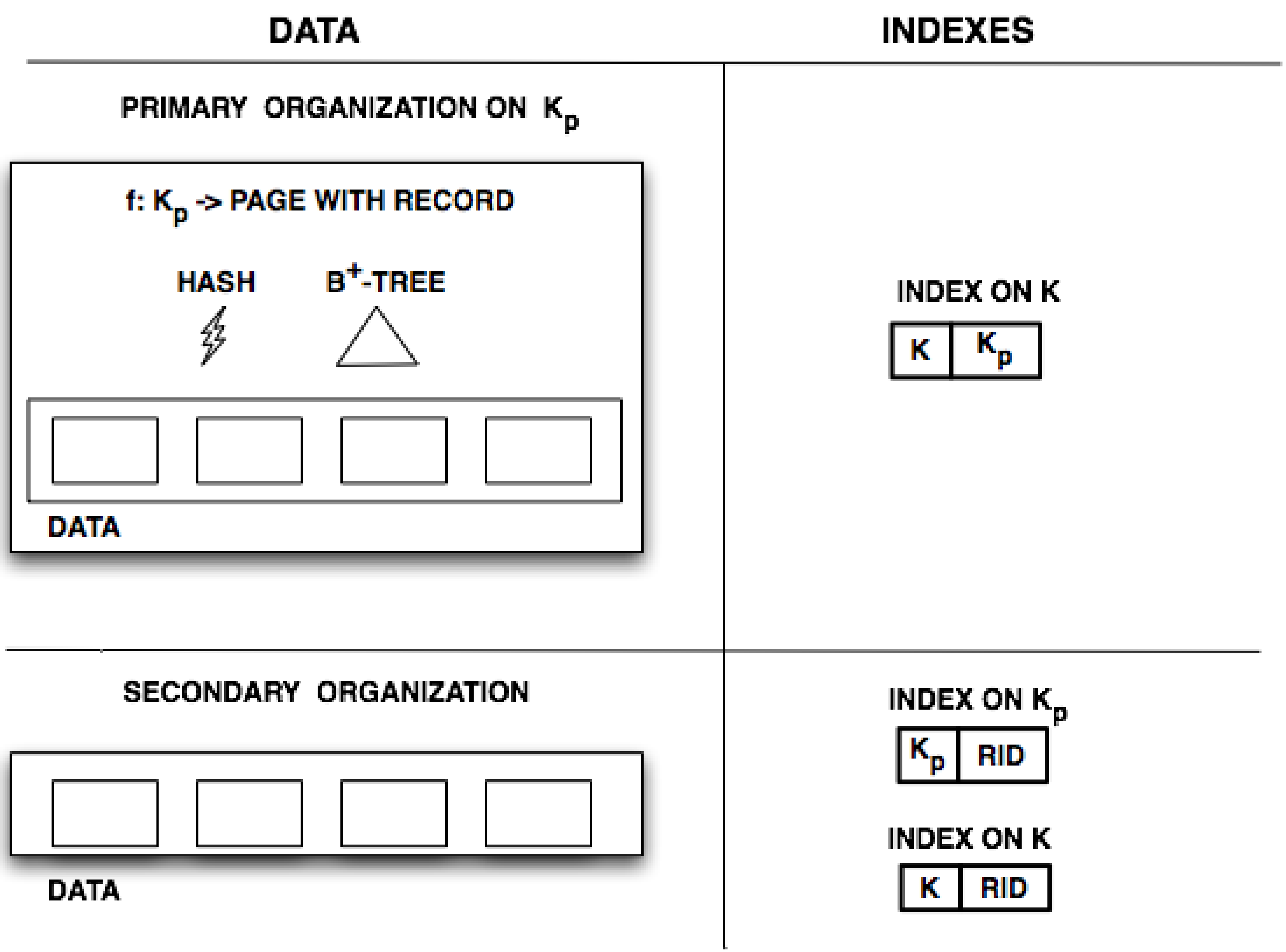
Index on **BirthYear**

BirthYear	RID
1970	2
1970	4
1970	5
1971	3
1972	1
1972	6

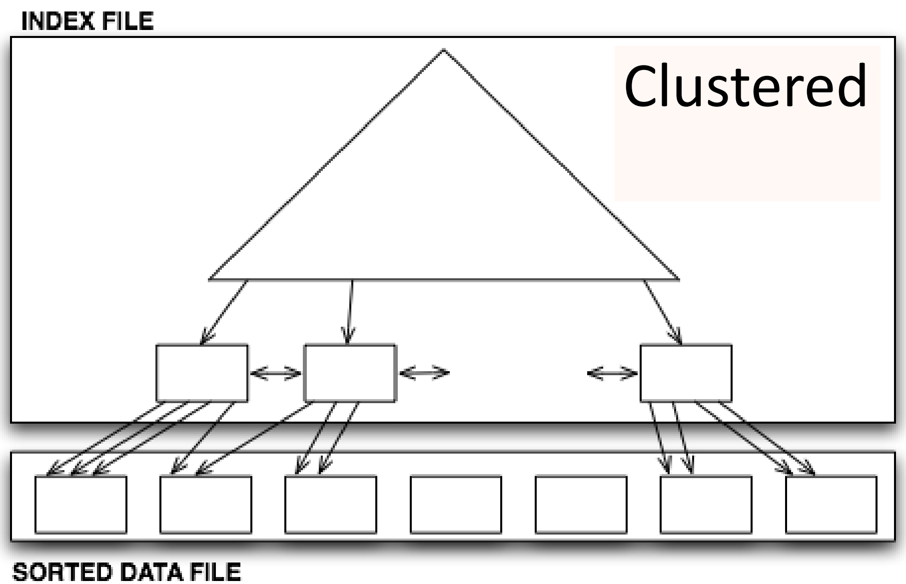
Clustered Indexes

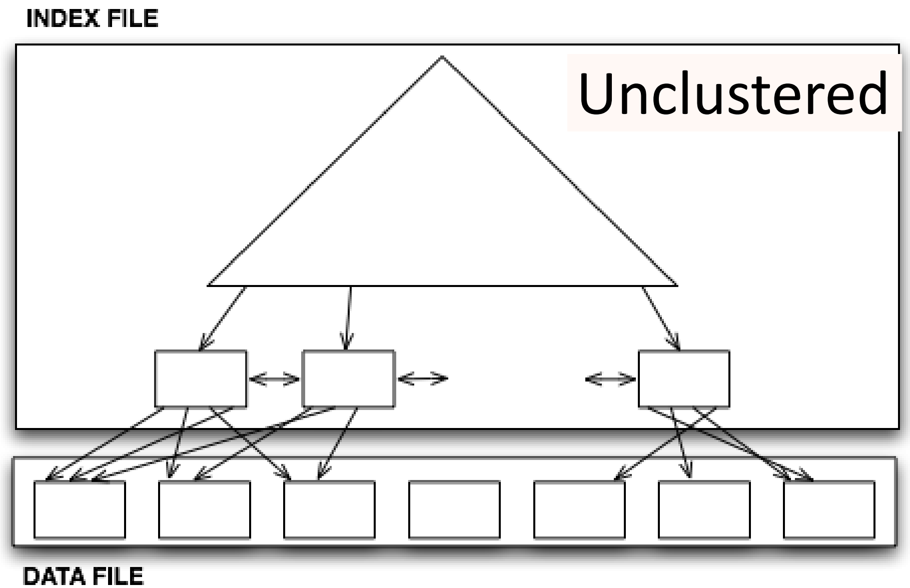
- Clustered vs. unclustered
- If the order of data records is the same as the order of data entries, then it is called clustered index.
- Clustered = with data almost ordered, if there are insertions

Data organizations for two keys: K_p and K



Clustered vs. Unclustered





Search cost	$C = CI + CD$
Equality search cost (k = v1)	$C = 1 + 1$
Range search cost ($p = (v1 \leq k \leq v2)$)	$\mathbf{sf}(p) = (v2 - v1)/(k_{max} - k_{min})$

$C_{clustered} = \mathbf{sf}(p) * N_{leaf} + \mathbf{sf}(p) * N_{pag}$

$C_{unclustered} = \mathbf{sf}(p) * N_{leaf} + \mathbf{sf}(p) * N_{rec}$

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