Indexed Sequential Files

COL 362 & COL 632

Indexing

01 Mar 2023

Insertion

Dense

Biology	
Comp. Sci.	
Elec. Eng.	
Finance	
History	
Music	
Physics	

76766	Crick	Biology	72000	
10101	Srinivasan	Comp. Sci.	65000	
45565	Katz	Comp. Sci.	75000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	
12121	Wu	Finance	90000	
76543	Singh	Finance	80000	
32343	El Said	History	60000	
58583	Califieri	History	62000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
33465	Gold	Physics	87000	

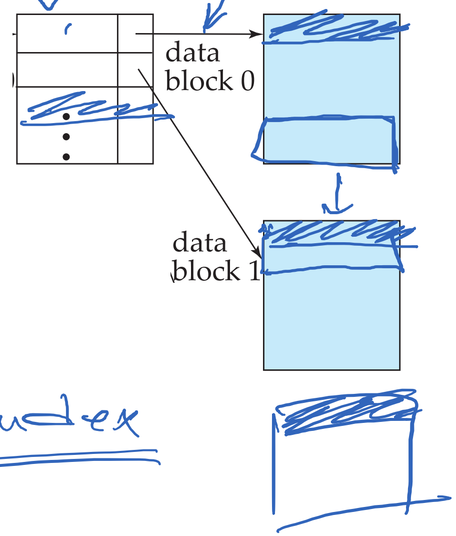
Energy Sciences

Overflow

Energy

Energy Material Science

Sparse



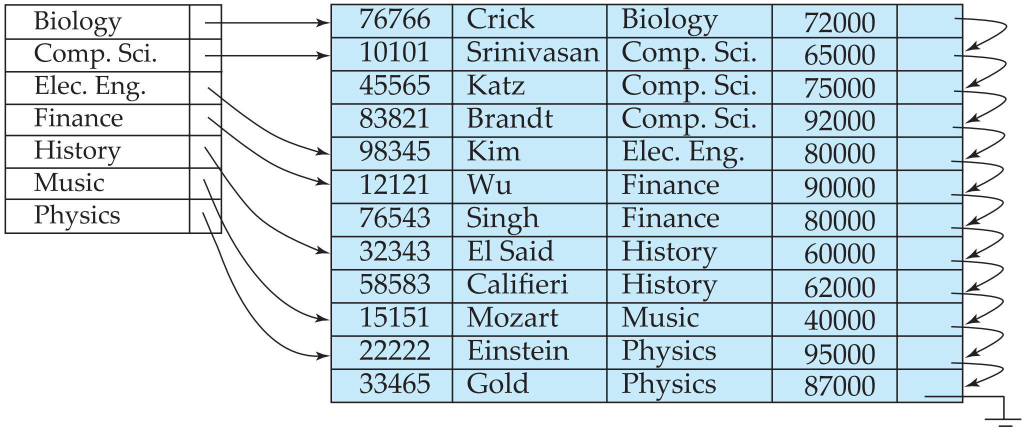
Sparse index

① Space is in the page

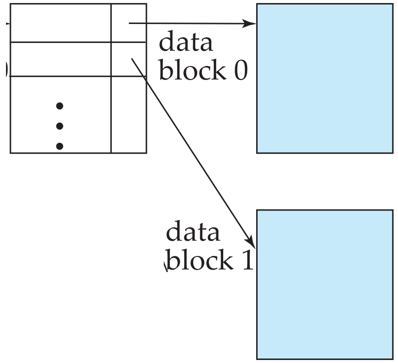
② create new page

Deletion

H. W.



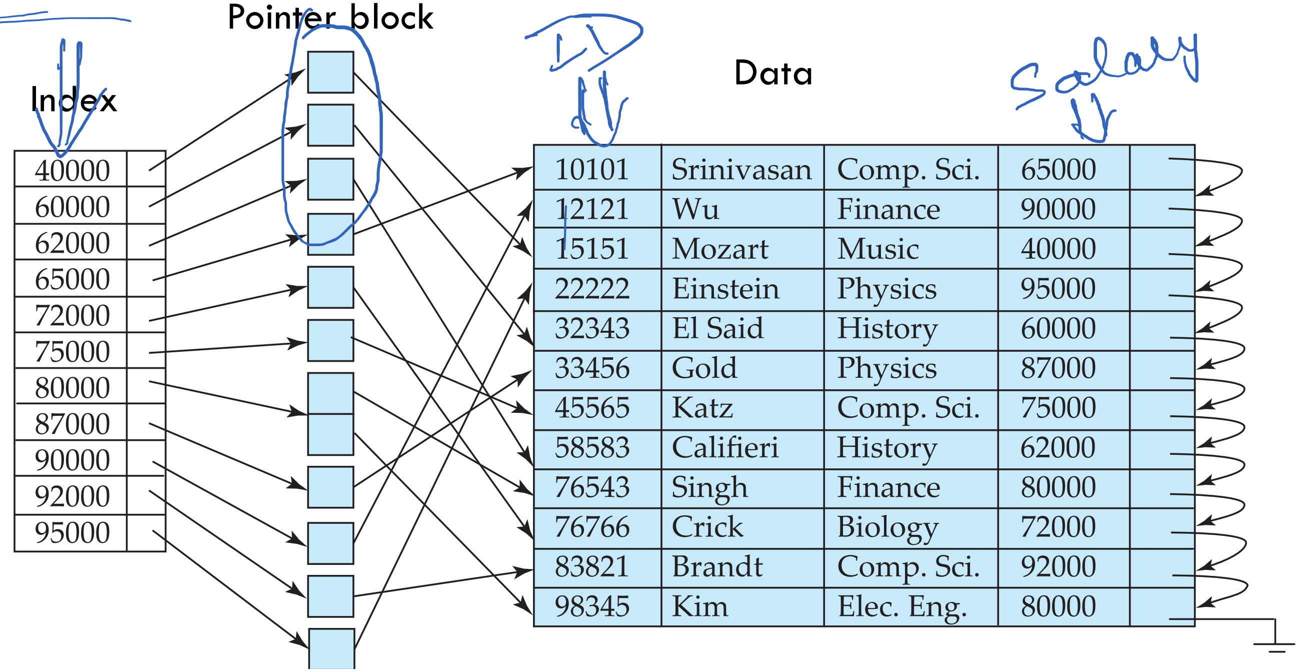
Biology
Comp. Sci.
Elec. Eng.
Finance
History
Music
Physics
76766 Crick Biology 72000
10101 Srinivasan Comp. Sci. 65000
45565 Katz Comp. Sci. 75000
83821 Brandt Comp. Sci. 92000
98345 Kim Elec. Eng. 80000
12121 Wu Finance 90000
76543 Singh Finance 80000
32343 El Said History 60000
58583 Califieri History 62000
15151 Mozart Music 40000
22222 Einstein Physics 95000
33465 Gold Physics 87000



data
block 0
data
block 1

Secondary Indexes

- Index record points to a bucket that contains pointers to all the actual records with that particular search-key value.
- Secondary indexes have to be dense



|| Goetz Graefe's monograph

B+-trees

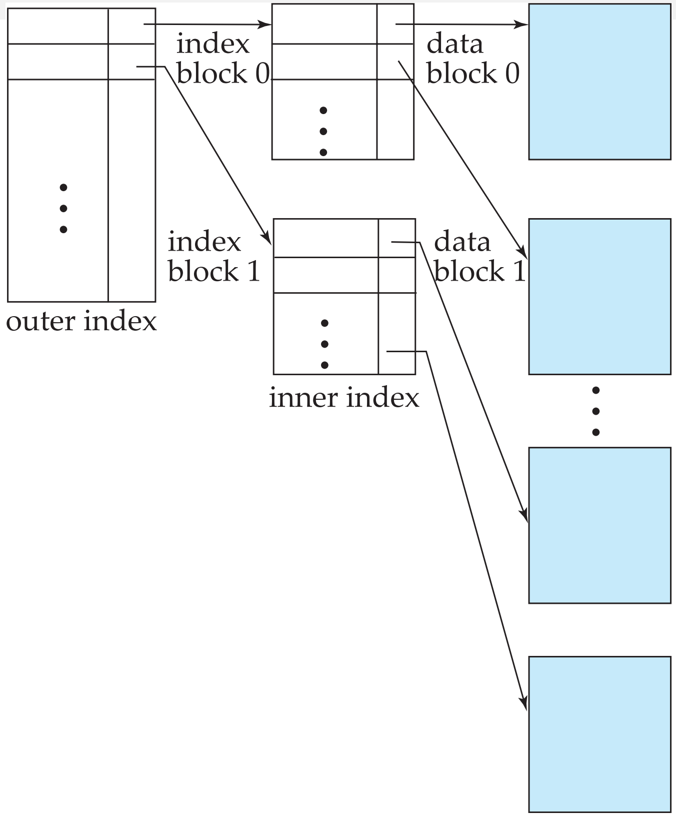
B^+-Trees (1/2)

Ubiqutous

(2,3) -trees +++

(a,b)

- Disadvantage of indexed-sequential files
  - performance degrades as file grows, since many overflow blocks get created
  - Periodic reorganization of entire file is required
- Advantage of B^+-tree index files
  - automatically reorganizes itself
  - Reorganization of entire file is not required
- (Minor) disadvantage of B^+-trees
  - extra insertion and deletion overhead, space overhead



B+-Tree (2/2)

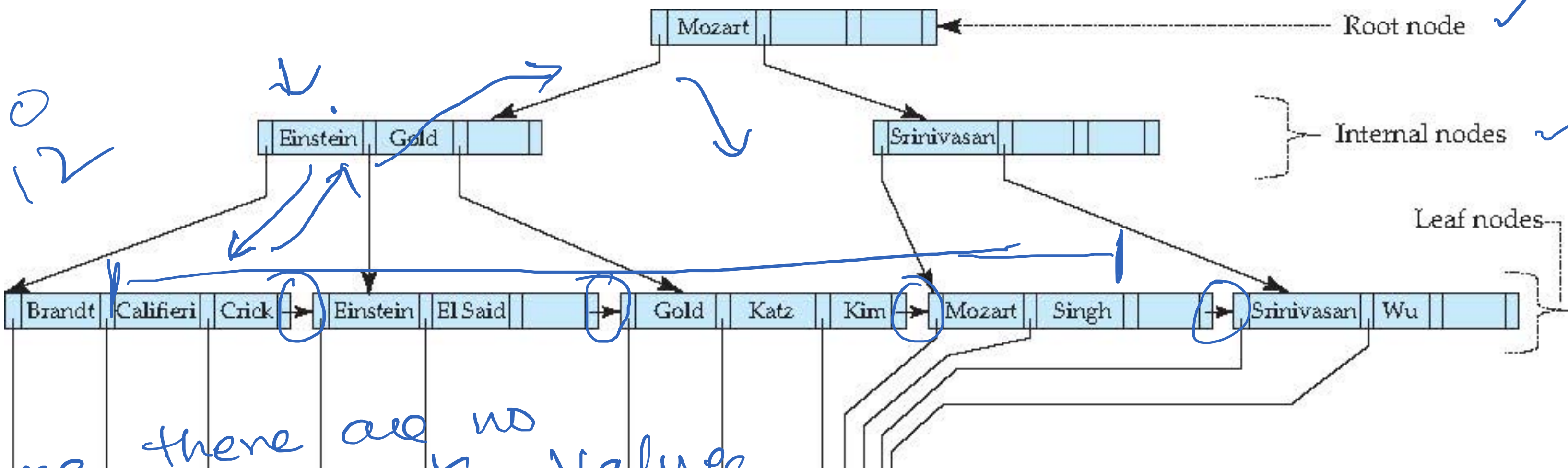
$\log_{\lceil n/2 \rceil} N$

$\lceil \frac{n}{2} \rceil \leq$ # of children $\leq n$

4

6

512



10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	80000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	60000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

data pointers

Assume there are no duplicates in key values

B⁺-Tree Node Structure

P_1	K_1	P_2	…	P_{n-1}	K_{n-1}	P_n

- K_i are the search-key values
- P_i are pointers to children (for non-leaf nodes) or pointers to records or buckets of records (for leaf nodes).
- The search-keys in a node are ordered

$$\boxed{K_1 < K_2 < K_3 < \ldots < K_{n-1}}$$

(Assuming no duplicate keys)

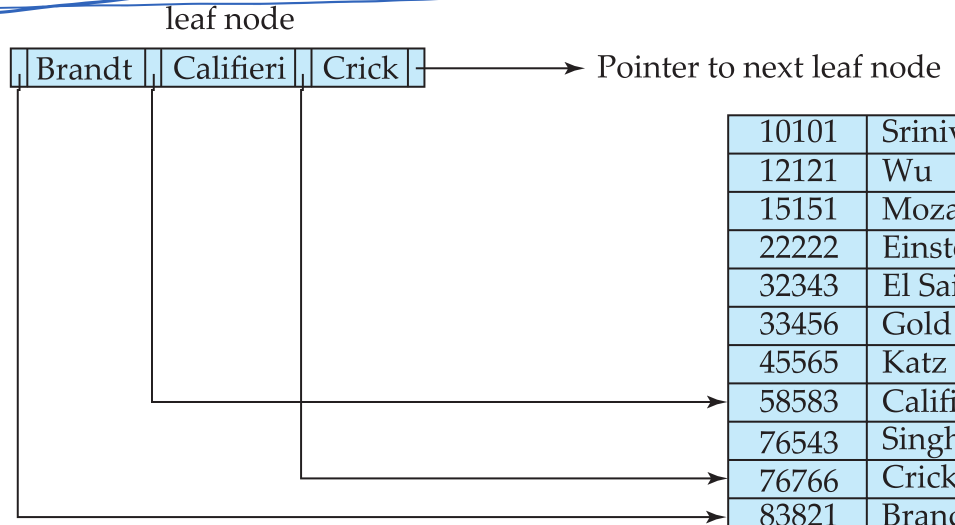
Block-linked list

Leaf Nodes in B$^+$-Trees

look like dense index

- For $i = 1, 2, \ldots, \boxed{n-1}$, pointer $\boxed{P_i}$ points to a file record with search-key value K_i,
- If L_i, L_j are leaf nodes and $i < j$, L_i's search-key values are less than or equal to L_j's search-key values
- P_n points to next leaf node in search-key order

P_n



10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	80000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	60000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

$\lceil n/2 \rceil$ $\lceil n \rceil$

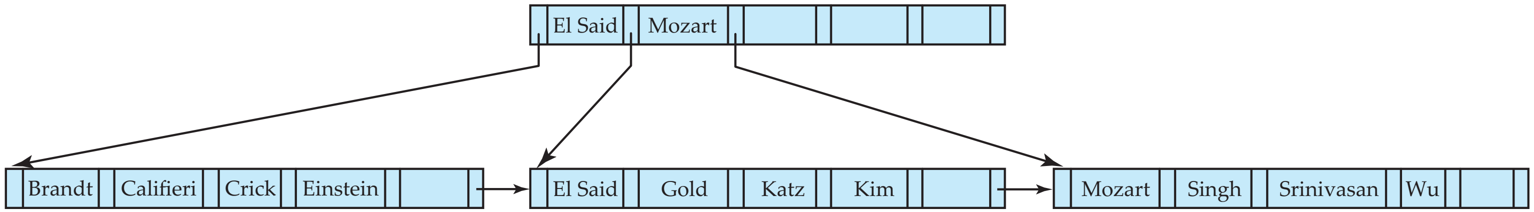
Non-Leaf Nodes in B$^+$-Trees

- For a non-leaf node with *m* pointers
  - All the search-keys in the subtree to which P_1 points are less than K_1
  - For $2 \leq i \leq n - 1$, all the search-keys in the subtree to which P_i points have values greater than or equal to K_{i-1} and less than K_i
  - All the search-keys in the subtree to which P_n points have values greater than or equal to K_{n-1}

K_2

P_1	K_1	P_2	$\ldots$	P_{n-1}	K_{n-1}	P_n

Example of B$^+$-tree



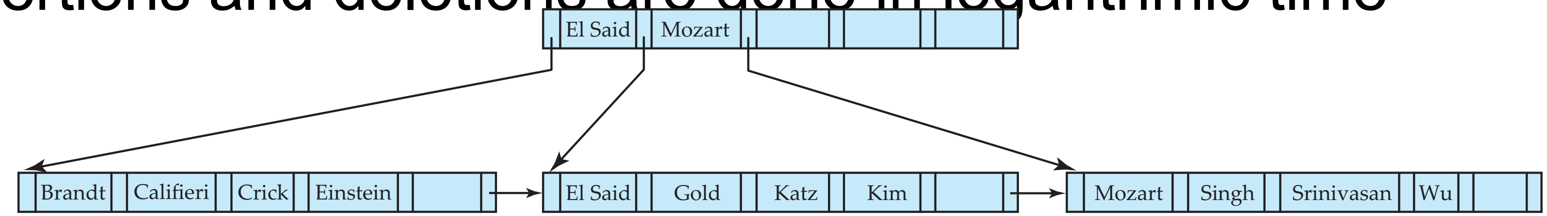
B$^+$-tree for `instructor` file (n = 6)

- Leaf nodes must have between 3 and 5 values ($\lceil (n-1)/2 \rceil$ and n –1, with n = 6).
- Non-leaf nodes other than root must have between 3 and 6 children ($\lceil (n/2 \rceil$ and n with n =6).
- Root must have at least 2 children.

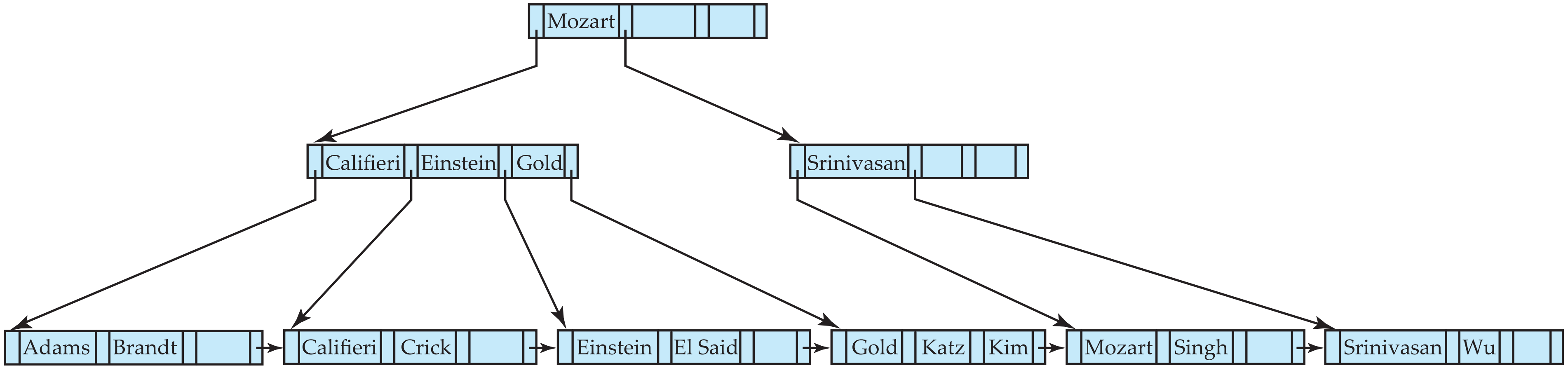
$n = 4$
6

Observations about B⁺-trees

- Since the inter-node connections are done by pointers, "logically" close blocks need not be "physically" close.
- The non-leaf levels of the B⁺-tree form a hierarchy of sparse indices.
- The B⁺-tree contains a relatively small number of levels
  - "Short and fat"
- Insertions and deletions are done in logarithmic time



Queries on B^+-Trees



Find record with search-key value *V*.

1. *C=root*
2. While C is not a leaf node {
   1. Let *i* be least value s.t. $V \leq K_i$.
   2. If no such exists, set *C* = *last non-null pointer in C*
   3. Else { if ($V = K_i$) Set $C = P_{i+1}$ else set $C = P_i$}

   }
3. Let *i* be least value s.t. $K_i = V$
4. If there is such a value *i*, follow pointer P_i to the desired record.
5. Else no record with search-key value *V* exists.

Analysis of B$^+$-trees

- No. of search keys: 1,000,000
- Block size: 4K
- Size of an index entry: 40B
- Max. no. of search keys per block:
- Max. height of the tree:
- No. of block accesses per query: