USN :

CMR Institute of Technology, Bangalore
DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
III - INTERNAL ASSESSMENT

Semester: 6-CBCS 2018
Subject: FILE STRUCTURES (18IS61)
Faculty: Mr Prasad B S

Date: 1 Aug 2021
Time: 09:00 AM - 10:30 AM
Max Marks: 50

Instructions to Students :

Answer any 5 full questions , each carries 10 marks.

Answer any 5 question(s)

Q.No		Marks	CO	PO	BT/CL
1	What are the properties of B-Tree? Explain deletion, merging and redistribution of elements on B-Tree with suitable example	10	CO3	PO1,PO2,PO3,PO4,PO5	L2
	: For a Btree of order m, Btree has the following properties. 1. Every page has a maximum of m children. 2. Every page, except for the root and the leaves, has a minimum of m/2 children 3. The root has a minimum of 2 children (unless it is a leaf). 4. All of the leaves are on the same level 5. The leaf level forms a complete, ordered index of the associated data file. The rules for deleting a key *k* from a node *n* in a B-tree are as follows: 1. If *n* has more than the minimum number of keys and the *k* is not the largest in *n*, simply delete *k* from *n*. 2. If n has more than the minimum number of keys and the *k* is the largest in *n*, delete *k* and modify the higher level indexes to reflect the new largest key in *n*. 3. If *n* has exactly the minimum number of keys and one of the siblings of *n* has few enough keys, merge *n* with its sibling and delete a key from the parent node. 4. If *n* has exactly the minimum number of keys and one of the siblings of *n* has extra keys, redistribute by moving some keys from a sibling to *n*, and modify the higher level indexes to reflect the new largest keys in the affected nodes. PS: Add example for deletion, merging and redistribution				
2	What is indexed sequential access? Explain the block splitting and merging due to insertion and deletion in sequence set.	10	CO2,CO3	PO1,PO2,PO3,PO4,PO5	L2
	: In indexed and tree structure based access user had to choose between viewing a file from an indexed point of view or from a sequential point of view. In Indexed sequential access we are looking for a single organizational method that provides both of these views simultaneously. **Sequence set:** • A sequence set is a set of records in physical key order which is such that it stays ordered as records are added and deleted. • Since sorting and resorting the entire sequence set as records are added and deleted is expensive, we look at other strategies. In particular, we look at a way to localize the changes. • The idea is to use blocks that can be read into memory and rearranged there quickly. Like in B-Trees, blocks can be split, merged or their records re-distributed as necessary.				

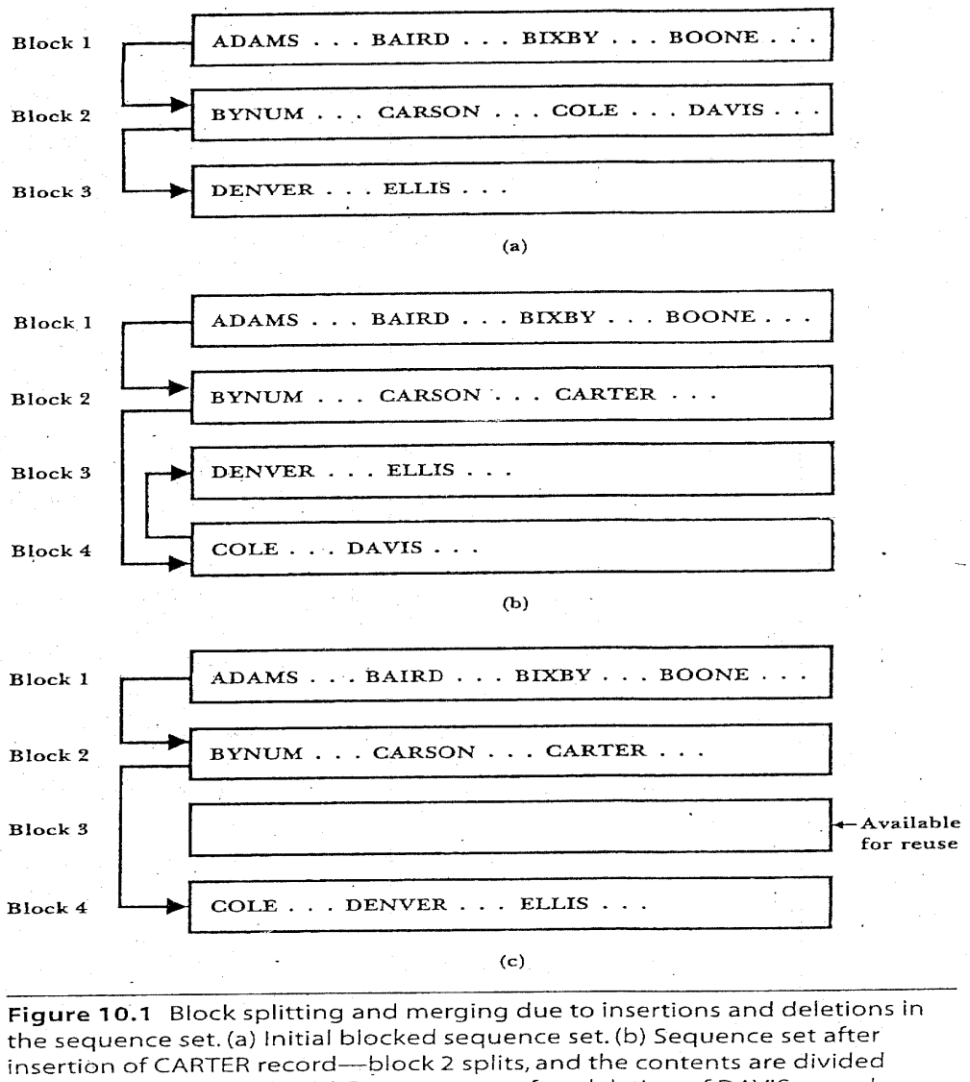
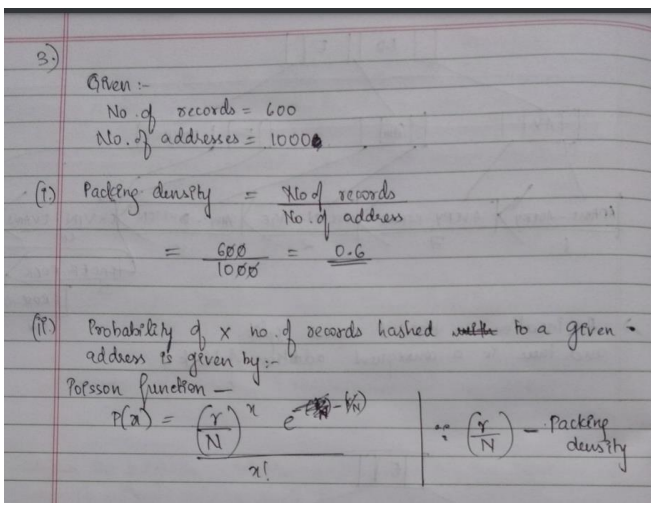
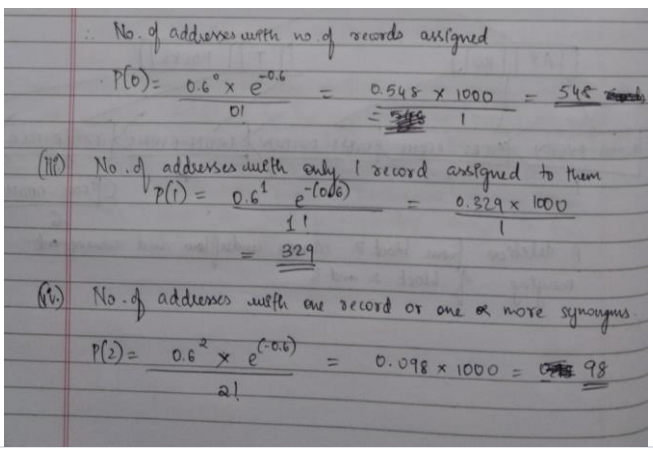
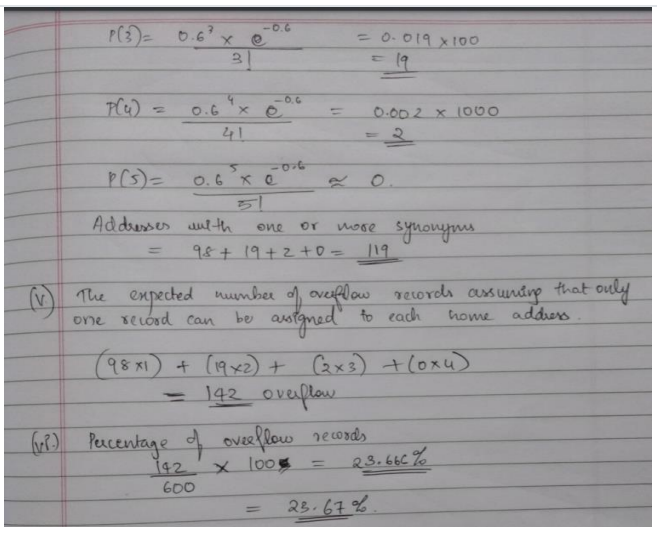


Figure 10.1 Block splitting and merging due to insertions and deletions in the sequence set. (a) Initial blocked sequence set. (b) Sequence set after insertion of CARTER record—block 2 splits, and the contents are divided between blocks 2 and 4. (c) Sequence set after deletion of DAVIS record—block 4 is less than half full, so it is concatenated with block 3.

3	Suppose that 1000 addresses are allocated to hold 600 records in a randomly hashed file and that each address can hold one record. Compute the following values: I. The packing density II. The expected number of addresses with no records assigned to them. III. The expected number of addresses with exactly one record assigned. IV. The expected number of addresses with one record or one or more synonyms V. The expected number of overflow records assuming that only one record can be assigned to each home addresses VI. Percentage of overflow records	10	CO3	PO1,PO2,PO3,PO4,PO5	L3

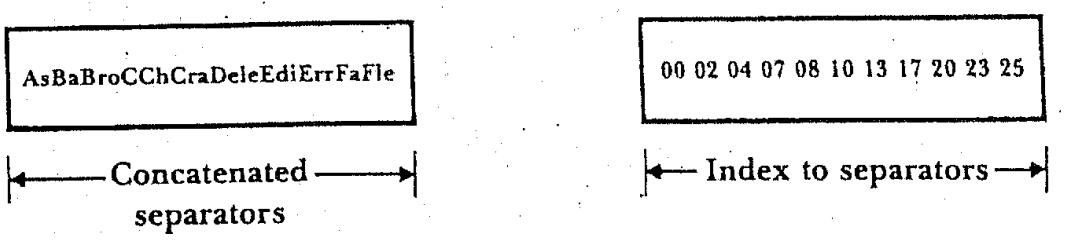






4	With a suitable diagram, explain the internal structure of index set blocks.	10	CO3	PO1,PO2,PO3,PO4,PO5	L2

To place the following Set of separators into an index block:
As, Ba, Bro, C, Ch, Cra, Pele, Edi, Err, Fa, Fe
Merge these separators and build an index for them, as shown in Fig.



The index set block needs some way to store references to its children, to the blocks descending from it in the next lower level of the tree. Assuming that the references are made in terms of a relative block number (RBN). If there are N separators within a block, the block has N+ 1 children and therefore needs Space to store N+ 1 RBNs in addition to the separators and the index to the separators.
One possible approach is illustrated in Fig below. In addition to the vector of separators, the index to these separators, and the list of associated block numbers, this block structure includes:
Separator count: we need this to help us find the middle element in the index to the separators so we can begin our binary search.
Total length of separators: the list of merged separators varies in length from block to block. Since the index to the separators begins at the end of this variable-length list, we need to know how long the list is so we can find the beginning of our index.

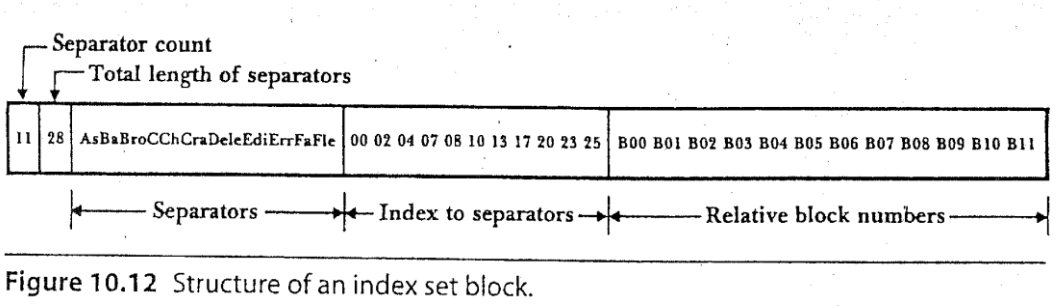
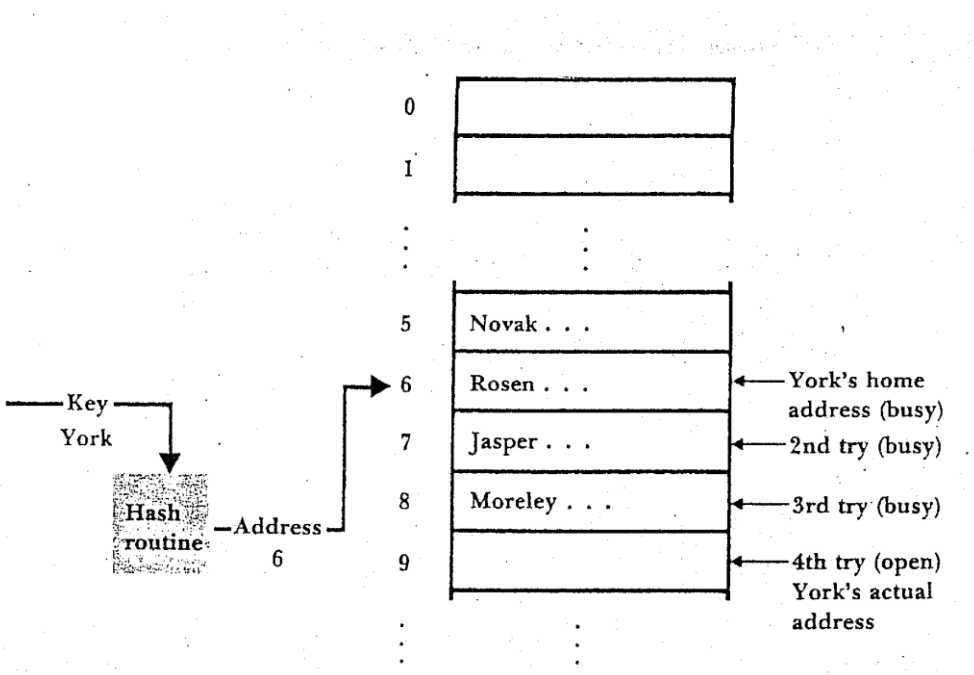


Figure 10.12 Structure of an index set block.

5	Define collision. Explain the different collision resolution techniques used in hashing.	10	CO3	PO1,PO2,PO3,PO4,PO5	L2

Collision: An attempt to store a record at an address which does not have sufficient room ie already occupied by another record which is a synonym. A collision occurs when two record keys has to the same address.
Progressive overflow: If a key, k1, hashes into the same address, a1, as another key, k2, then look for the first available address, a2, following a1 and place k1 in a2. If the end of the address space is reached, then wrap around it. When searching for a key that is not in, if the address space is not full, then an empty address will be reached or the search will come back to where it began.

a. Progressive Overflow causes extra searches and thus extra disk accesses. If there are many collisions, then many records will be far from "home". **Buckets:** Buckets allows



to store more than one record in a single hash address i.e. storing More than One Record per address. A bucket describes a block of records sharing the same address that is retrieved in one disk access. When a record is to be stored or retrieved, its home bucket address is determined by hashing.

Effect of bucket on collision:

To compute how densely packed a file is, we need to consider
1) the number of addresses, N, (buckets)
2) the number of records we can put at each address, b, (bucket size) and
3) the number of records, r.
Then,
Packing Density = r/bN

Though the packing density does not change when halving the number of addresses and doubling the size of the buckets, the expected number of overflows decreases dramatically.

Disadvantage: When a bucket is filled, we still have to worry about the record overflow problem, but this occurs much less often than when each address can hold only one record.

b. Double Hashing:

Double hashing is similar to progressive overflow. The second hash value is used as a stepping distance for the probing. The second hash value should never be one. The second hash value should be relatively prime to the size of the table. \

If there is collision at the hash address produced by h1(k) = Say 'X', then the key is dropped
to second hash function to produce address 'C'. The new record is stored at the address 'X+C'. A collision resolution scheme which applies a second hash function to keys which collide, to determine a probing distance 'C'.

h1(k)= X (collision at X)
h2(k)= C
X+C = home address of k

The use of double hashing will reduce the average number of probes required to find a record.

c. Chained Progressive Overflow: Chained progressive overflow forms a linked list, or chain, of synonyms. Each home address contains a number indicating the location of the next record with the same home address. The next record in turn contains a pointer to the other record with the same home address. Assume the following has address generation:

Key	Home address
Adams	20
Bates	21
Cole	20
Deans	21
Evans	24
Flint	20

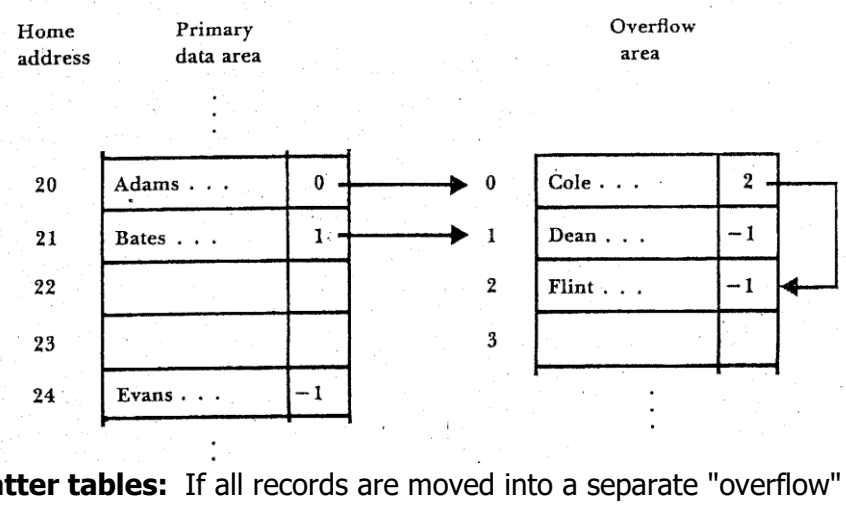
The records with same home address are linked as shown in the figure below: In the figure below Adams contain the pointer to Cole which is synonym. Then Bates contain pointer to Dean which are again synonym.

Address	Key	Next record
20	Adams	22
21	Bates	23
22	Cole	25
23	Deans	-1
24	Evans	-1
25	Flint	-1
26		
.		
.		

d. Chaining with a separate overflow area:

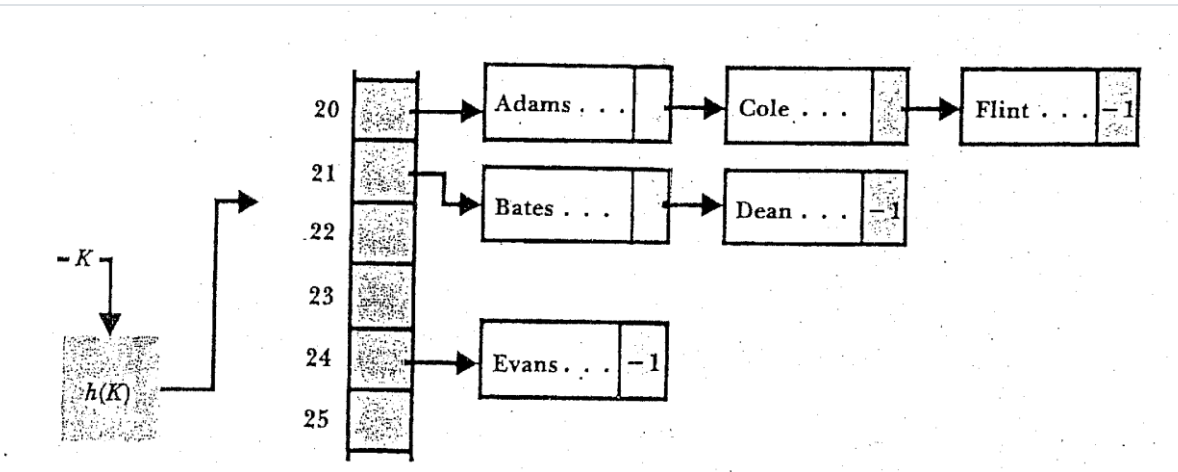
One way to keep overflow records from occupying home addresses where they should not be is to move them all to a separate overflow area. The set of home addresses is called prime data area, and the set of overflow addresses is called the overflow area.

Whenever a new record is added if its home address is empty, it is stored in primary storage area. Otherwise it is moved to overflow area, where it is added to a linked list that starts at home address.



e. Scatter tables: If all records are moved into a separate "overflow" area, with only links being left in the hash table, the result is a scatter table. In scatter table records are not stored at the hash address. Only pointer to the record is stored at hash address.

Scatter tables are similar to index but they are searched by hashing rather than some other method which is used in case of indexing. Main advantage of scatter tables is they support use of variable length records.



6	Explain and illustrate the working of Simple Prefix B+ Tree with suitable example. What are the issue in maintenance of Simple prefix B+ Tree.	10	CO3	PO1,PO2,PO3,PO4,PO5	L2

B-tree index set taken together with sequence set forms a file structure called a simple prefix B+ tree. The modifier simple prefix indicates that the index set contains shortest separators or prefixes of the keys rather than copies of the actual key.

Maintenance:

1. **Changes Localized to Single Blocks in the Sequence Set**

   Additions, deletions, and updates in the sequence set which affect only a single block do not affect the index set.

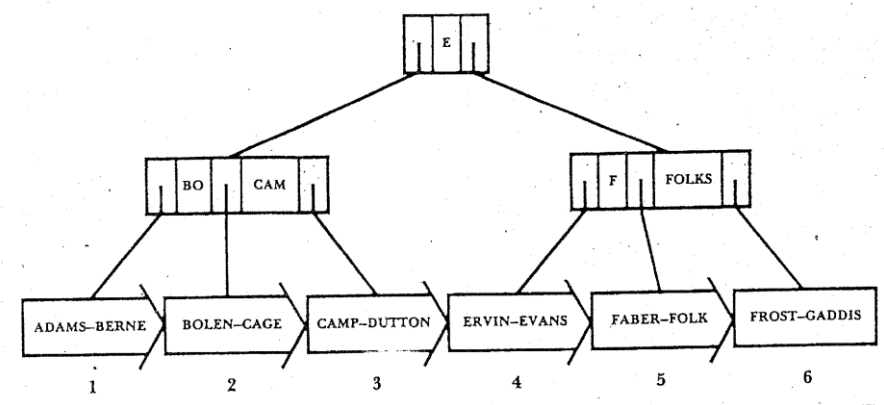


Figure 10.8 The deletion of the EMBRY and FOLKS records from the sequence set leaves the index set unchanged.

2. **Changes involving multiple blocks in the sequence set**

   When addition to the sequence set results in split in the sequence set, deletion in sequence set which results in merger, or changes in sequence set resulting in redistribution requires involvement of more than one block set and corresponding changes in the index set as well.

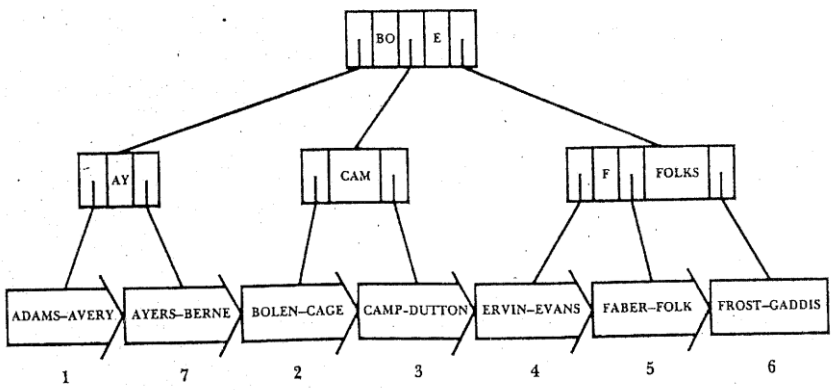


Figure 10.9 An insertion into block 1 causes a split and the consequent addition of block 7. The addition of a block in the sequence set requires a new separator in the index set. Insertion of the AY separator into the node containing BO and CAM causes a node split in the index set B-tree and consequent promotion of BO to the root.

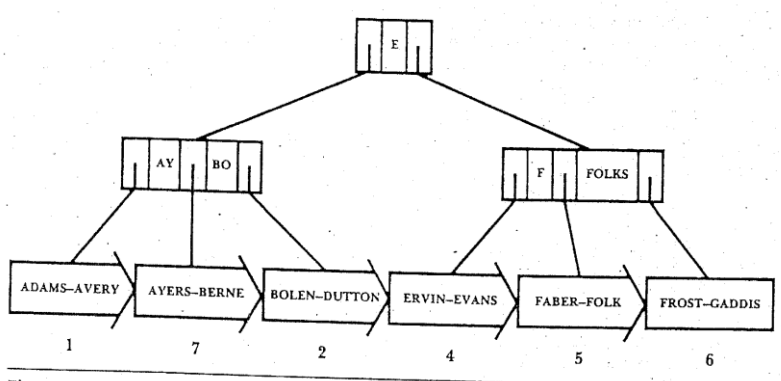


Figure 10.10 A deletion from block 2 causes underflow and the consequent merging of blocks 2 and 3. After the merging, block 3 is no longer needed and can be placed on an avail list. Consequently, the separator CAM is no longer needed. Removing CAM from its node in the index set forces a merging of index set nodes, bringing BO back down from the root.

7	Write a note on the following: 1. Worst Case Search Depth of B -Tree 2. B* Trees	10	CO2,CO3	PO1,PO2,PO4,PO5	L2

For a B-tree of order m, the minimum number of descendants from the root page is 2, so the second level of the tree contains only 2 pages. Each of these pages, in turn, has at least $\lceil m/2 \rceil$ descendants. The third level, then, contains

$$2 \times \lceil m/2 \rceil$$

pages. Since each of these pages, once again, has a minimum of $\lceil m/2 \rceil$ descendants, the general pattern of the relation between depth and the minimum number of descendants takes the following form:

Level	Minimum number of descendants
1 (root)	2
2	$2 \times \lceil m/2 \rceil$
3	$2 \times \lceil m/2 \rceil \times \lceil m/2 \rceil$ or $2 \times \lceil m/2 \rceil^2$
4	$2 \times \lceil m/2 \rceil^3$
...	...
d	$2 \times \lceil m/2 \rceil^{d-1}$

So, in general, for any level d of a B-tree, the *minimum* number of descendants extending from that level is

$$2 \times \lceil m/2 \rceil^{d-1}$$

For a tree with N keys in its leaves, we can express the relationship between keys and the minimum height d as

$$N \geq 2 \times \lceil m/2 \rceil^{d-1}$$

Solving for d, we arrive at the following expression:

$$d \leq 1 + \log_{\lceil m/2 \rceil} (N/2).$$

This expression gives us an *upper bound* for the depth of a B-tree with N keys. Let's find the upper bound for the hypothetical tree that we describe at the start of this section: a tree of order 512 that contains 1 000 000 keys. Substituting these specific numbers into the expression, we find that

$$d \leq 1 + \log_{256} 500\,000$$

or

$$d \leq 3.37$$

So we can say that given 1 000 000 keys, a B-tree of order 512 has a depth of no more than three levels.

Forming a B* tree

Consider a system in which we are postponing splitting through redistribution, as outlined in the preceding section. If we are considering any page other than the root, we know that when it is finally time to split, the page has at least one sibling that is also full. This opens up the possibility of a two-to-three split rather than the usual one-to-two or two-way split.

The important aspect of this two-to-three split is that it results in pages that are each about two-thirds full rather than just half full. This makes it possible to define a new kind of B-tree, called a B* tree, which has the following properties:

1. Every page has a maximum of m descendants.
2. Every page except for the root has at least $\lceil (2m - 1)/3 \rceil$ descendants.
3. The root has at least two descendants (unless it is a leaf).
4. All the leaves appear on the same level.