INDEXING

An index is a collection of data entries which is used to locate a record in a file. Index table record in a file consist of two parts, the **first part** consists of value of prime or non-prime attributes of file record known as **indexing field** and, the **second part** consists of a pointer to the location where the record is physically stored in memory. In general, index table is like the index of a book, that consists of the name of topic and the page number. During searching of a file record, index is searched to locate the record memory address instead of searching a record in secondary memory. On the basis of properties that affect the efficiency of searching, the indexes can be classified into **two** categories.

1. Ordered indexing
2. Hashed indexing

Ordered Indexing

In ordered indexing, records of file are stored in some sorted order in physical memory. The values in the index are ordered (sorted) so that binary search can be performed on the index. Ordered index can be divided into two categories.

1. Dense indexing
2. Sparse indexing

Dense and Sparse Indexing

- *Dense index :* In dense indexing there is a record in index table for each unique value of the search-key attribute of file and a pointer to the first data record with that value. The other records with same value of search-key attribute are stored sequentially as shown in Figure 1

Advantages of Dense index

(i) It is efficient technique for small and medium sized data file.

(ii) Searching is comparatively fast and efficient.

Disadvantages of Dense index

(i) Index table is large and require more memory space.
(ii) Insertion and deletion is comparatively complex.
(iii) In-efficient for large data files.

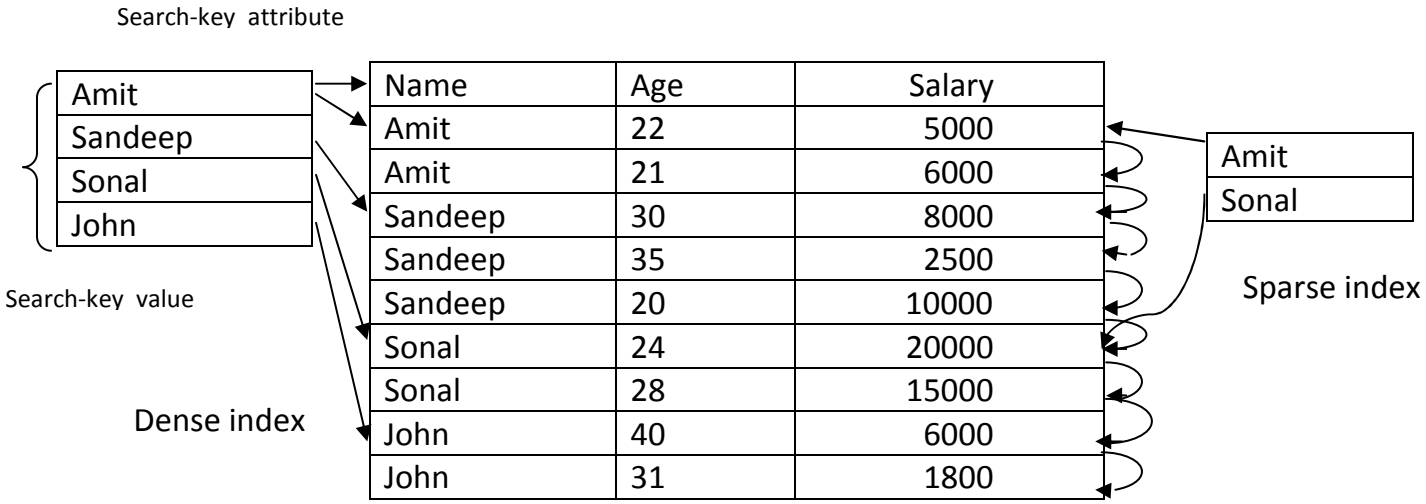


FIGURE 1. *Dense and sparse index.*

- Sparse index : On contrary, in sparse indexing there are only some records in index table for unique values of the search-key attribute of file and a pointer to the first data record with that value. To search a record in sparse index we search for a value that is less than or equal to value in index for which we are looking. After getting the first record, linear search is performed to retrieve the desired record. There is at most one sparse index since it is not possible to build sparse index that is not clustered.

Advantages of sparse index

(i) Index table is small and hence save memory space (especially in large files).
(ii) Insertion and deletion is comparatively easy.

Disadvantages of sparse index

(i) Searching is comparatively slower, since index table is searched and then linear search is performed inside secondary memory.

Clustered and Non-Clustered Indexes

- *Clustered index:* In clustered, index file records are stored physically in order of non-prime key attribute that does not have a unique value for each record. The non-prime key field is known as clustering field and index is known as clustering index. It is same as dense index. A file can have at most one clustered index as it can be clustered on at most one search key attribute. It may be sparse.
- *Non-Clustered index:* An index that is not clustered is known as non-clustered index. A data file can have more than one non-clustered index.

Primary and Secondary Index

- *Primary index:* A primary index consists of all prime-key attributes of a table and a pointer to physical memory address of the record of data file. To retrieve a record on the basis of all primary key attributes, primary index is used for fast searching. Binary search is done on index table and then directly retrieve that record from physical memory. It may be sparse.

Advantages of Primary index

(i) Search operation is very fast.
(ii) Index table record is usually smaller.
(iii)) A Primary index is guaranteed not to duplicate.

Disadvantages of Primary index

(i) There is only one primary index of a table. To search a record on less than all prime-key attributes, linear search is performed on index table.
(ii) To create a primary index of an existing table, record should be in some sequential order otherwise database is required to be adjusted.

- *Secondary index : A secondary* index provides a secondary means of accessing a data file. A secondary index may be on a candidate key field or on non-prime key attributes of a table. To retrieve a record on the basis

of non-prime key attributes, secondary index can be used for fast searching. Secondary index must be dense with a index entry for every search key value and a pointer to every record in a file.

Advantages of secondary index

(i) Improve search time if search on non-prime key attributes.
(ii) A data file can have more than one secondary index.

Disadvantages of Secondary index

(i) A secondary index usually needs more storage space.
(ii) Search time is more than primary index.
(iii) They impose a significant overhead on the modification of database.

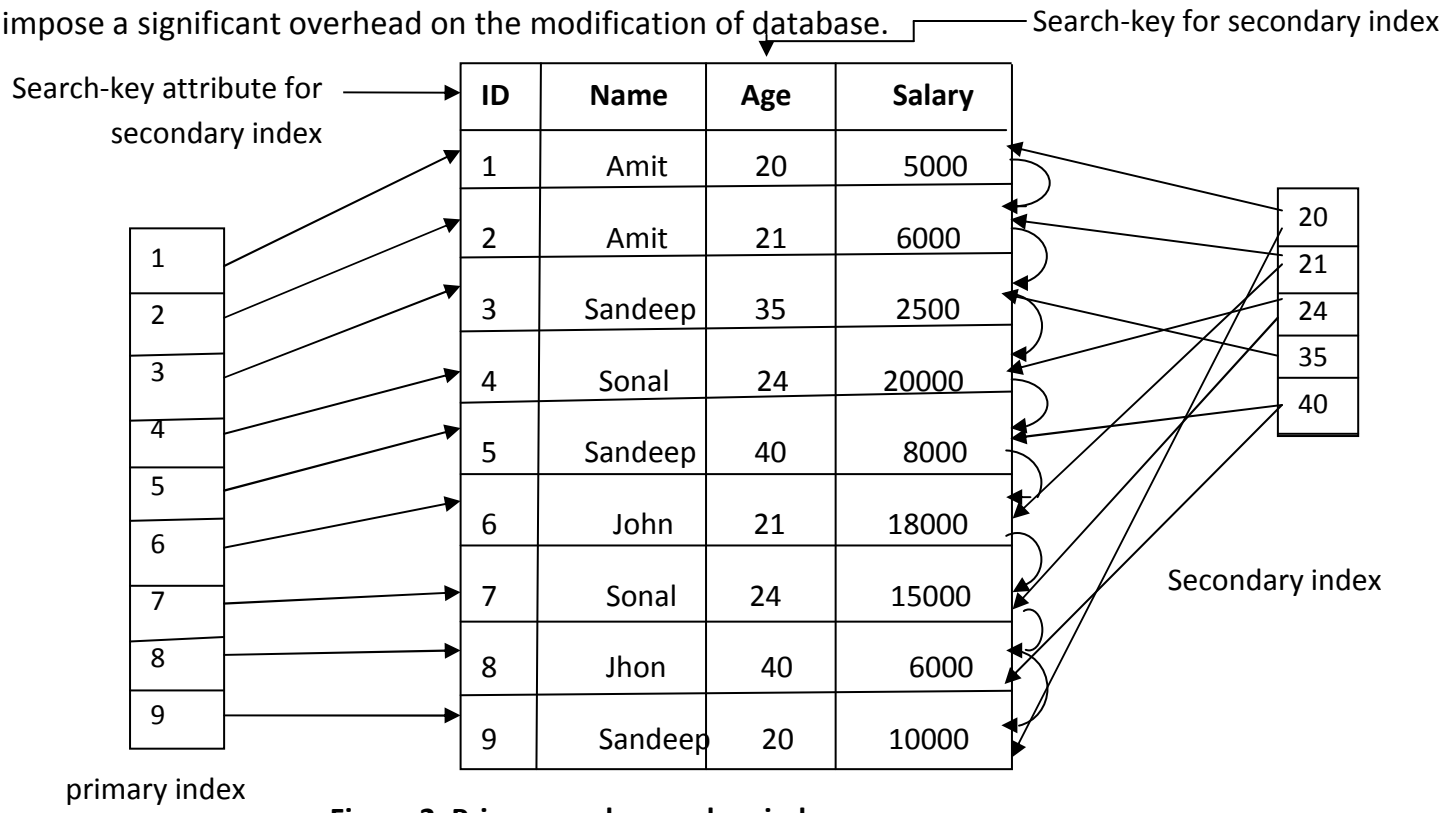


Figure 2. Primary and secondary index

Single and Multilevel Indexes

- *Single level indexes:* A single stage index for a data file is known as single level index.

  A single level index cannot be divided. It is useful in small and medium size data files. If the file size is bigger, then single level, indexing is not an efficient method. Searching is faster than other indexes for small size data files

- *Multilevel index :* A single index for a large size data file increases the size of index table and increases the search time that results in slower searches. The idea behind multilevel indexes is that, a single level index is divided into multiple levels, which reduces search time.

  In multilevel indexes, the first level index consists of two fields, the first field consists of a value of search key attributes and a second field consists of a pointer to the block (or second level index) which consists that value and so on.

  To search a record in multilevel index, binary search is used to find the largest of all the small value or equal to the one that needs to be searched. The pointer points to a block of the inner index. After reaching to the desired block, the desired record is searched (in case of two-level indexing) otherwise again the largest of the small values or equal to the one that needs to be searched and so no.

Benefits of multilevel indexes are they reduce search time significantly for large size data files.

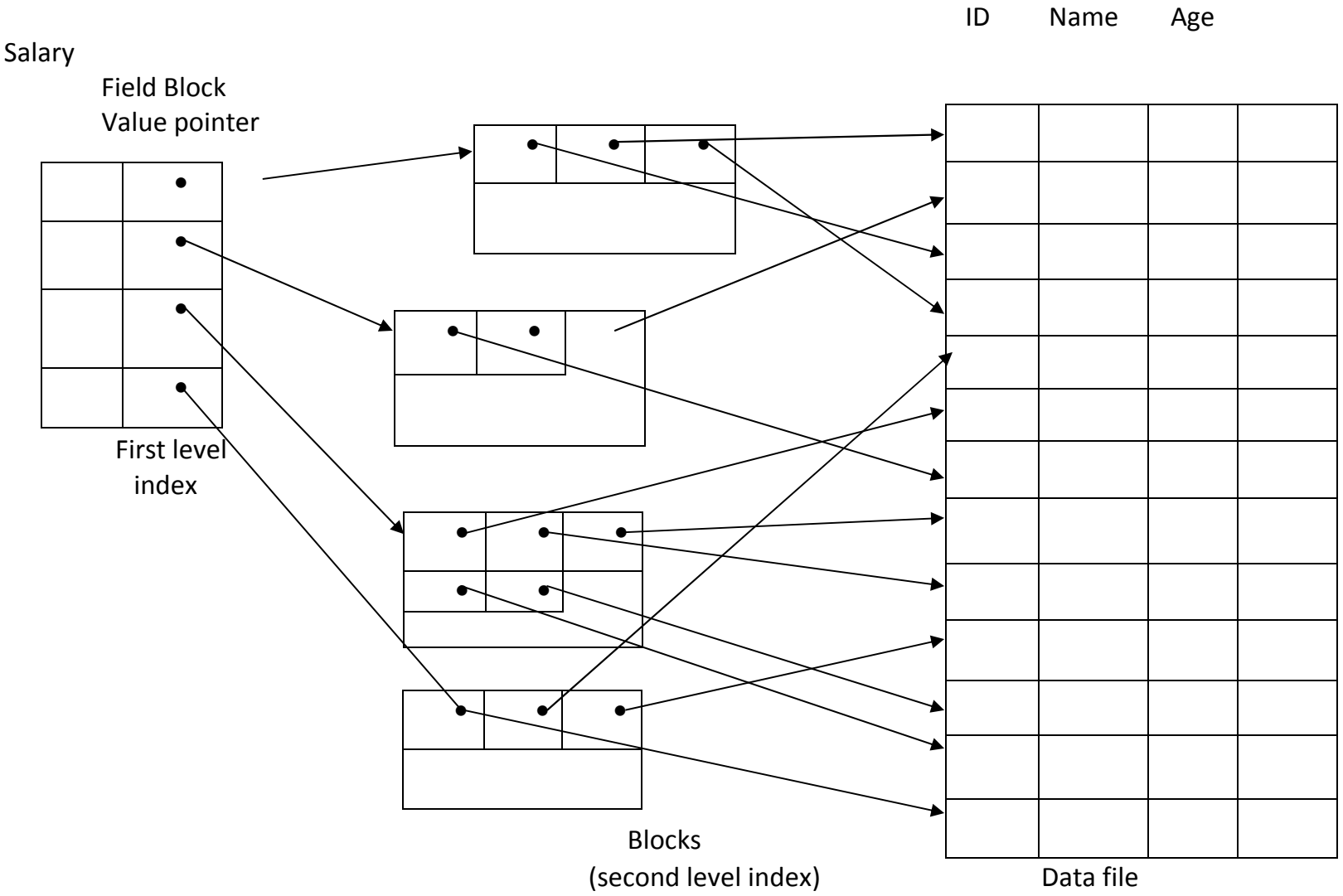


FIGURE 3. *Multilevel indexing*

Hashed Indexing

To overcome the disadvantage of ordered indexing,a hash index can be created for a data file. Hashing allow us to avoid accessing an index structure. A hashed index consists of two fields, the first field consists of search key attribute value and second field consists of pointer to the hash file structure. Hashed indexing is based on values of records being uniformly distributed using a hashed function.

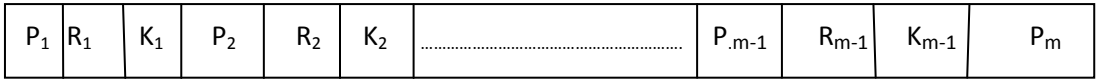
B-TREE INDEX FILES

In B-Tree index files, tree structure is used. A B-tree of order m is an m-way search tree with the following properties.

1. Each node of the tree, except the root and leaves, has at least $\lceil \frac{1}{2}n \rceil$ subtrees and no more than n subtrees. It ensures that each node of tree is at least half full.
2. The root of the tree has at least two subtrees, unless it is itself a leaf. It forces the tree to branch early.
3. All leaves of the tree are on the same level. It keeps the tree nearly balanced.

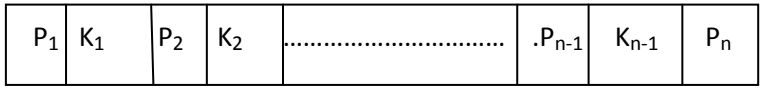
To overcome the performance degradation w.r.t. growth of files in index sequential files B-tree index files are used. It is a kind of multilevel index file organization. In tree structure search start from the root node and stops at leaf node.

(i) Non-Leaf Node :



In non-leaf node, there are two pointers. Pointer P_i points to any other node. Pointer R_i points to the block of actual records or storage area of records, K_i represents the value.

(ii) Leaf Node :



In leaf node, there is only one pointer P_i which points to block of actual records or storage area of records. K_i represents the key value. A B-tree for file employee is shown in Figure 4.

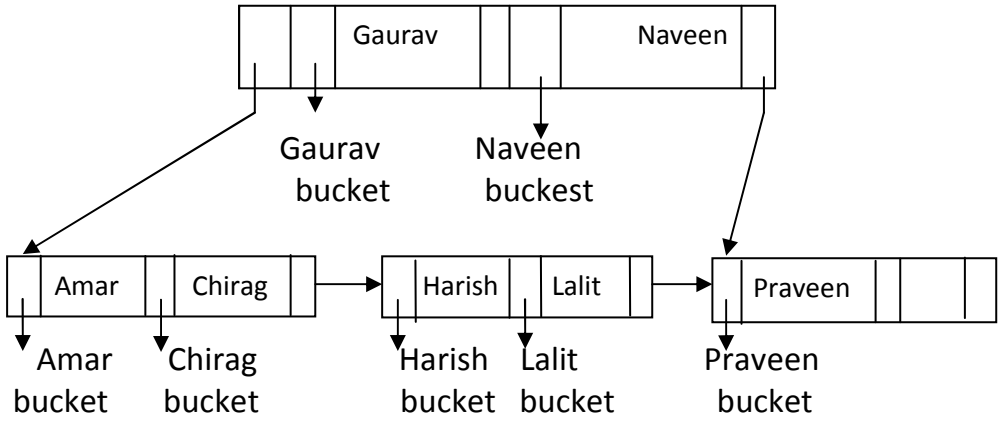


FIGURE 4. *B-tree for file Employee.*

Operations :

Searching a record : Searching a record with its key value starts from root node. It is possible to find desired record without going leaf node in B-tree.

Deletion of a record : Deletion in B-tree is a complicated process. If desired entry is in leaf node then, simply delete it otherwise find a proper replacement for that entry.

Insertion of a record : B-tree is a balanced tree and insertion of a new record in B-tree cause node splits and therefore affects the height of the tree.

Advantages

(i) Key value appears only once in the node.

(ii) Searching is faster than indexed sequential files.

(iii) Performance is maintained w.r.t. growth of file size.

Disadvantages

(i) Updation of records are more complicated then B^+ trees.

(ii) Less efficient than B^+ trees and direct files.

(iii) Searching time is still proportional to logarithm of the number of search keys.

B^+ - TREE INDEX FILES

A B+-tree is a kind of balanced tree. The length of every path from root of the tree to the leaf of the tree are same. The number of children n is fixed for a particular tree. Each non-leaf node in the tree has between (n/2) and n children. Index used in B^+-tree files is multilevel index but its structure differ from the index structure used in multilevel index-sequential files.

A typical node of B^+-tree contains up to n-1 search key values and pointers. K_i represents search key value and P_i represents pointer to a file record, as shown in Figure 5

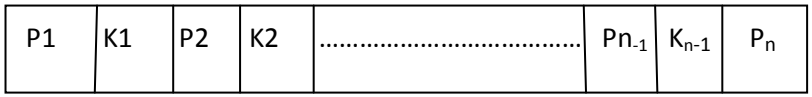


Figure 5. B^+- *tree for file employee*.

In B^+-tree search key values are in sorted order, thus , if $i < j$, then $K_i < K_j$. The number of pointer in a node is called Fanout of the node.

(a) Leaf nodes : In a leaf node, a pointer P_i points to either a file record having search key value K_i or to a bucket of pointers where each pointer to a file with search key value K_i (In case of secondary index in

which index is made of non-prime key attributes and files is not sorted in search key value order). A leaf node for file Employee (ID, Name, Salary) is shown in Figure 3.25

(ii) Non-leaf nodes : The structure of non-leaf nodes are same as of leaf node with a single difference that pointer P_i points to the tree nodes. It contains at least (n/2) pointers and a maximum of n pointers.

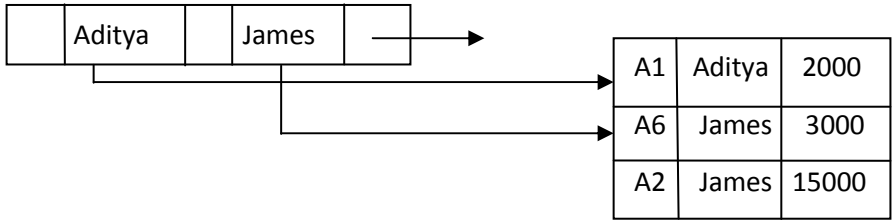


FIGURE 3.25. *A leaf node of B+-tree with index n = 3*

(iii) Root nodes : A root node contains at least 2 pointer and a maximum of less than [n/2] pointer. A B^+-tree contains only a single node if root node consists only a single pointer.

A pointer P_i with a search key value K_i in a non-leaf node points to a part of subtree having search key values less than K_i, and greater than or equal to K_{i-1}. pointer P_m points to a part of subtree having search key values greater than or equal to K_{m-1}, pointer P1 points to the part of subtree having search key values less than K_1.

A B^+-tree for file employee is shown in Figure 6

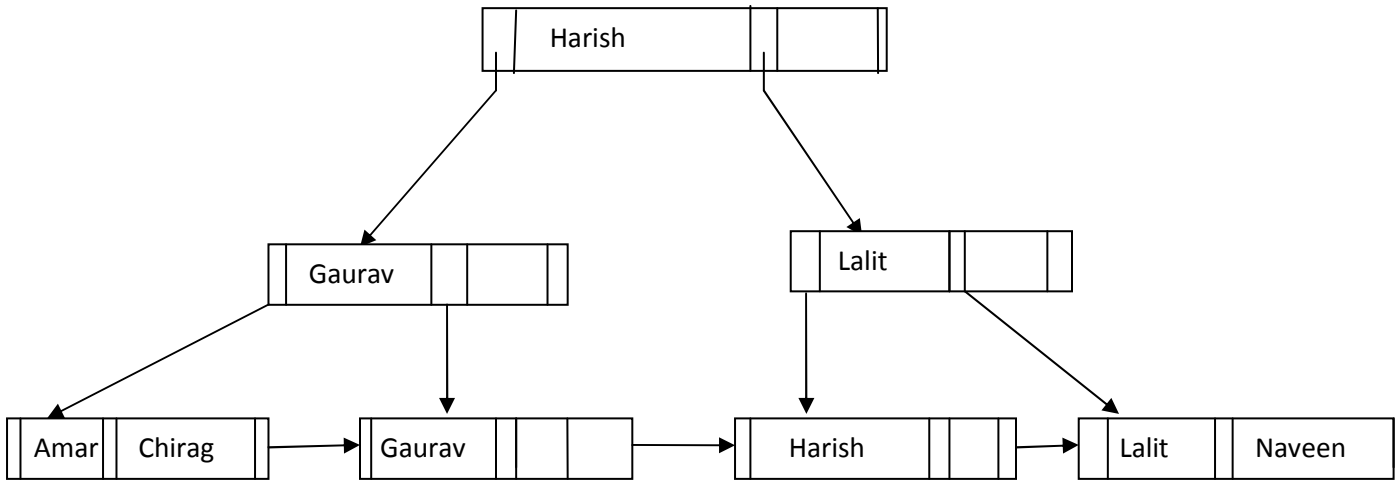


FIGURE 6. *B^+-tree for file employee*.

Searching a record : Searching a record with its key value starts from root node. It is possible to find desired record without going to leaf node in B^+-tree.

Deletion of a record : Deletion in B^+-tree is complicated process in some special cases. If desired entry is in leaf node then, simply delete it and if bucket is associated then bucket becomes empty as a result. If deletion causes a node to be less than half full, then it is combined with its neighboring nodes and it is propagated all the way to the root.

Insertion of a record : To insert a new record in B+-tree, first search a leaf node with same search key value. Simply add a new record to file or add a pointer in bucket, which points the record. If search key value does not

appear, simply insert the value in leaf node in correct position or create a new bucket with the appropriate pointer if necessary.

Advantage of B^+ -tree

(i) It provides a reasonable performance for direct access.

(ii) It provides an excellent performance foe sequential and range accesses.

(iii) Searching is faster.

Disadvantages of B^+ -trees

(i) Insertion is more complex than B-trees.

(ii) Deletion is more complex than B-tree.

(iii) Search key values are duplicated which results in wastage of memory space.

COMPARISON OF DIFFERENT FILE ORGANIZATIONS

S.No.	Sequential	Indexed	Hashed/Direct
1.	Random retrieval on primary key is impractical	Random retrieval of primary key is moderately fast.	Random retrieval of primary key is very fast.
2.	There is no wasted space for data.	No wasted space for data but there is extra space for index	Extra space for addition and deletion of records.
3.	Sequential retrieval on primary key is very fast.	Sequential retrieval on primary key is moderately fast.	Sequential retrieval of primary key is impractical
4.	Multiple key retrieval sequential file organization is possible.	Multiple key retrieval is very fast with multiple indexes.	Multiple key retrieval is not possible.
5.	Updating of records generally requires rewriting of file.	Updating of records generally requires maintenance of indexes.	Updating of records is the easiest one.
6.	Addition of new records requires rewriting the file.	Addition of new records is easy and requires maintenance of indexes.	Addition of new records is very easy.
7.	Deletion of records can create wasted space.	Deletion of records is easy if space can be allocated dynamically.	Deletion of records is very easy.

FACTORS AFFECTING CHOCE OF FILE ORGANIATION

(i) *Access type :* In order to search a record, whether random access or sequential access is required.

(ii) *Access time :* The total time taken by file organization to find a particular record.

(iii) *File size :* Choice is also dependent upon file size. If file size is large then choose direct access otherwise choose sequential access.

(iv) *Overhead :* Each technique has some overhead (it may be space overhead, time overhead etc). Any technique giving fast access may waste more space then slower techniques.

(v) *Updation time :* Time required to add, delete and modify any record also plays important role in efficiency.

(vi) *Complexity :* If technique is fast then it is complex and expensive. Whether funds are available to adopt new techniques.

(vii) *Availability of hardware :* The hardware that supports file organization. Example tape reader supports only sequential file organization