File Organizations and Indexing

File Organization

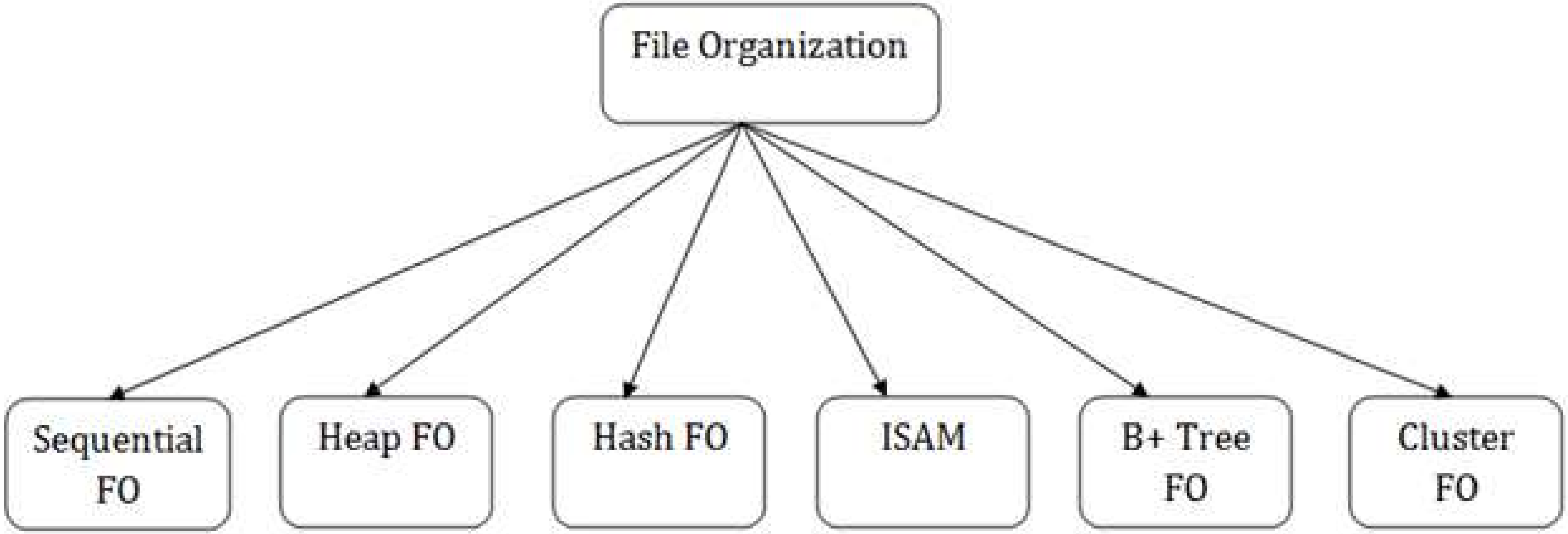
- The **File** is a collection of records. Using the primary key, we can access the records.
- File organization is a logical relationship among various records. This method defines how file records are mapped on to disk blocks.
- File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
- The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file.
- An alternative approach is to structure our files so that we can contain multiple lengths for records.
- Files of fixed length records are easier to implement than the files of variable length records.

Objective of file organization

- It contains an optimal selection of records, i.e., records can be selected as fast as possible.
- To perform insert, delete or update transaction on the records should be quick and easy.
- The duplicate records cannot be induced as a result of insert, update or delete.
- For the minimal cost of storage, records should be stored efficiently.

Types of file organization:

- File organization contains various methods. These particular methods have pros and cons on the basis of access or selection.
- In the file organization, the programmer decides the best-suited file organization method according to his requirement.



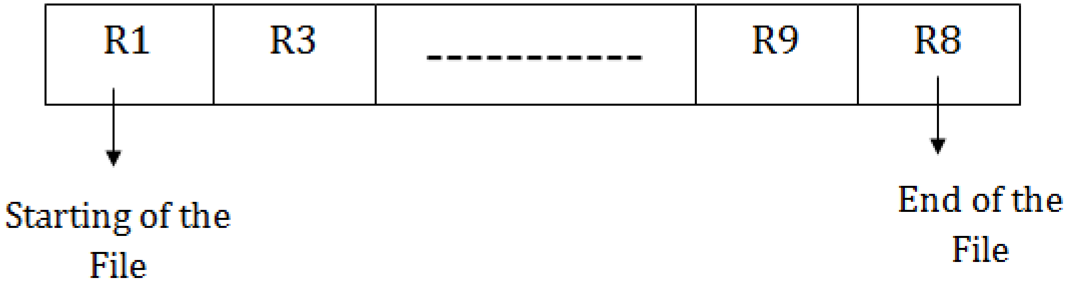
Sequential File Organization

- This method is the easiest method for file organization.
- In this method, files are stored sequentially.
- This method can be implemented in two ways:

- **1. Pile File Method**
- **2. Sorted File Method**

1. Pile File Method

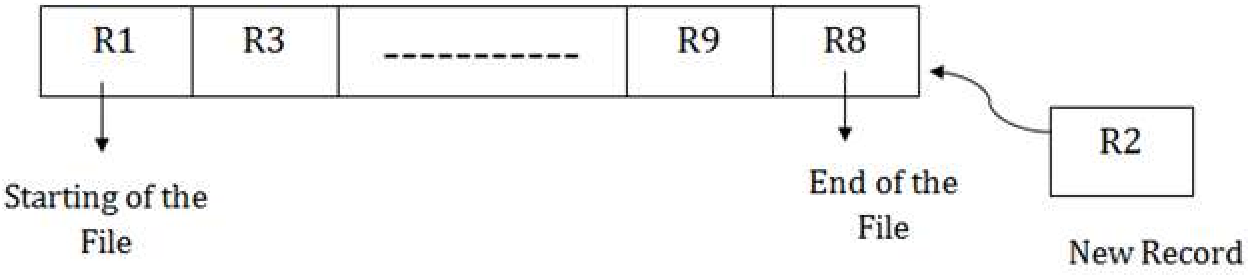
- It is a quite simple method. In this method, we store the record in a sequence, i.e., one after another. Here, the record will be inserted in the order in which they are inserted into tables.
- In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.



R1 R3 ----------- R9 R8

Starting of the File

End of the File



R1 R3 ----------- R9 R8

R2

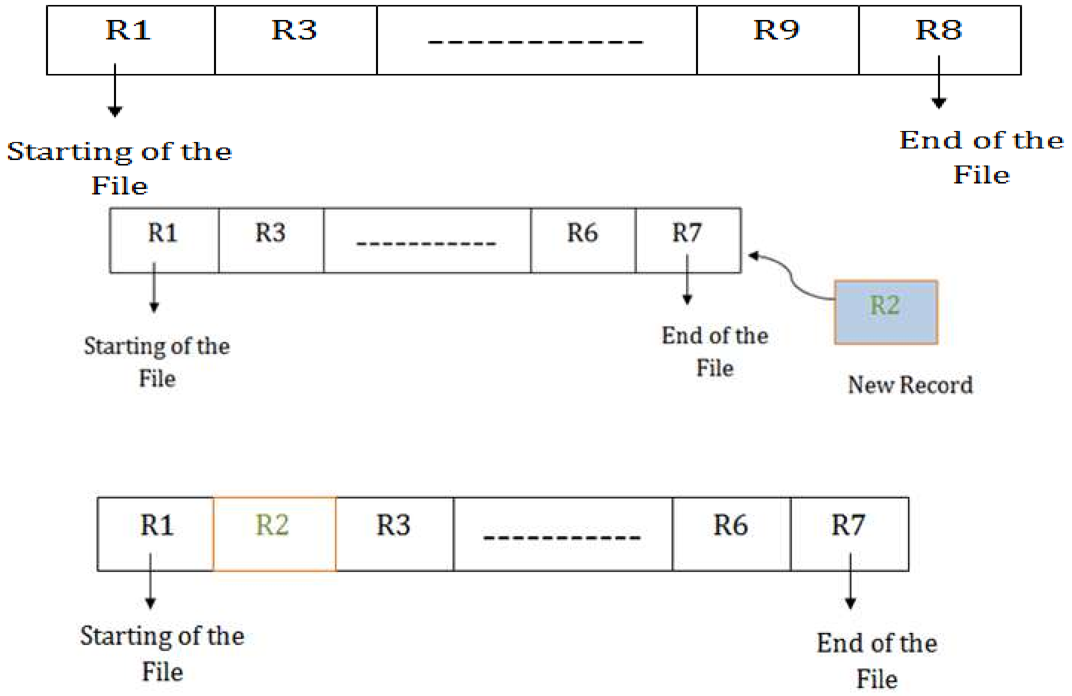
Starting of the File

End of the File

New Record

2. Sorted File Method

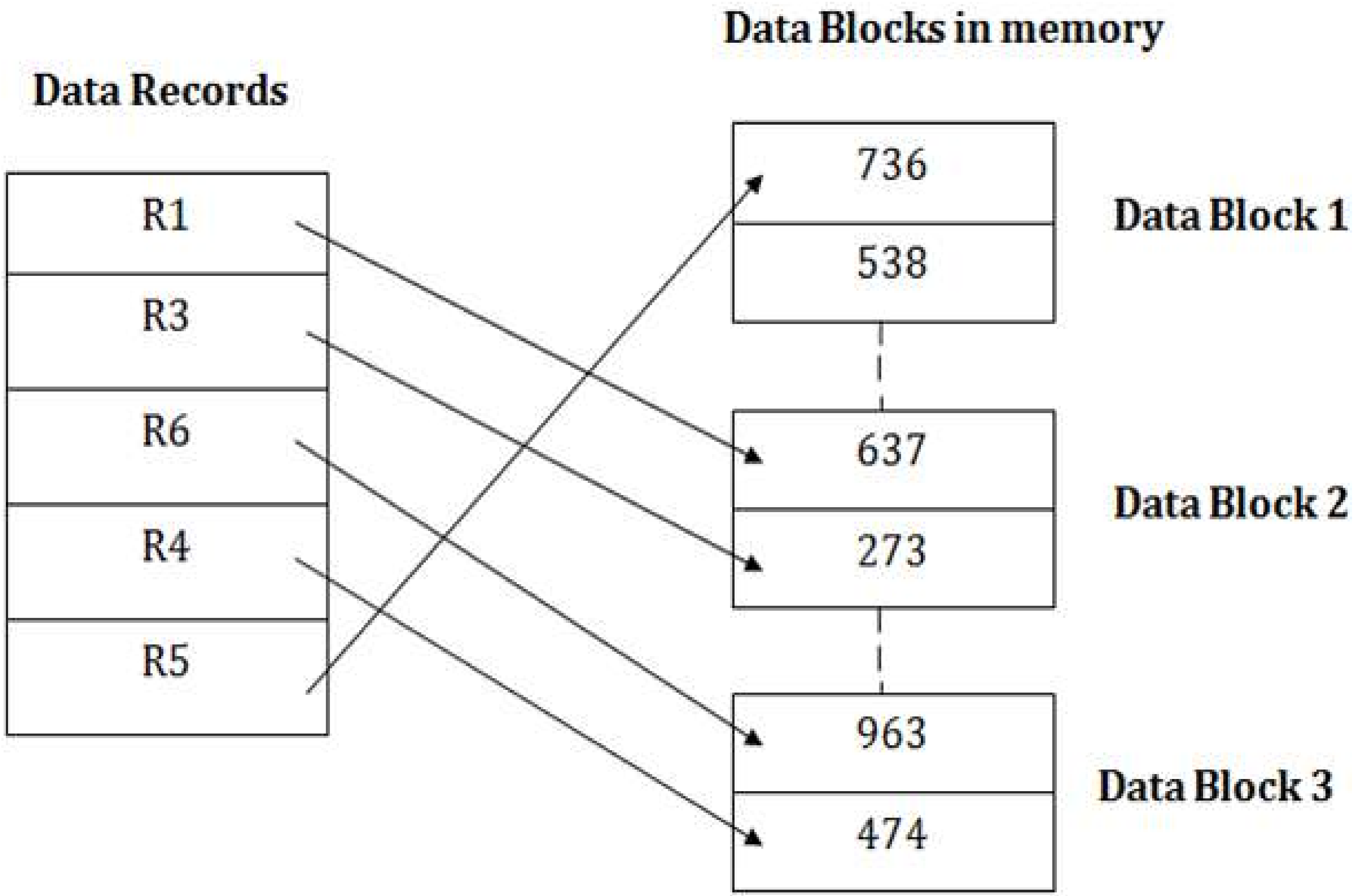
- In this method, the new record is always inserted at the file's end, and then it will sort the sequence in ascending or descending order.
- Sorting of records is based on any primary key or any other key.
- In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.



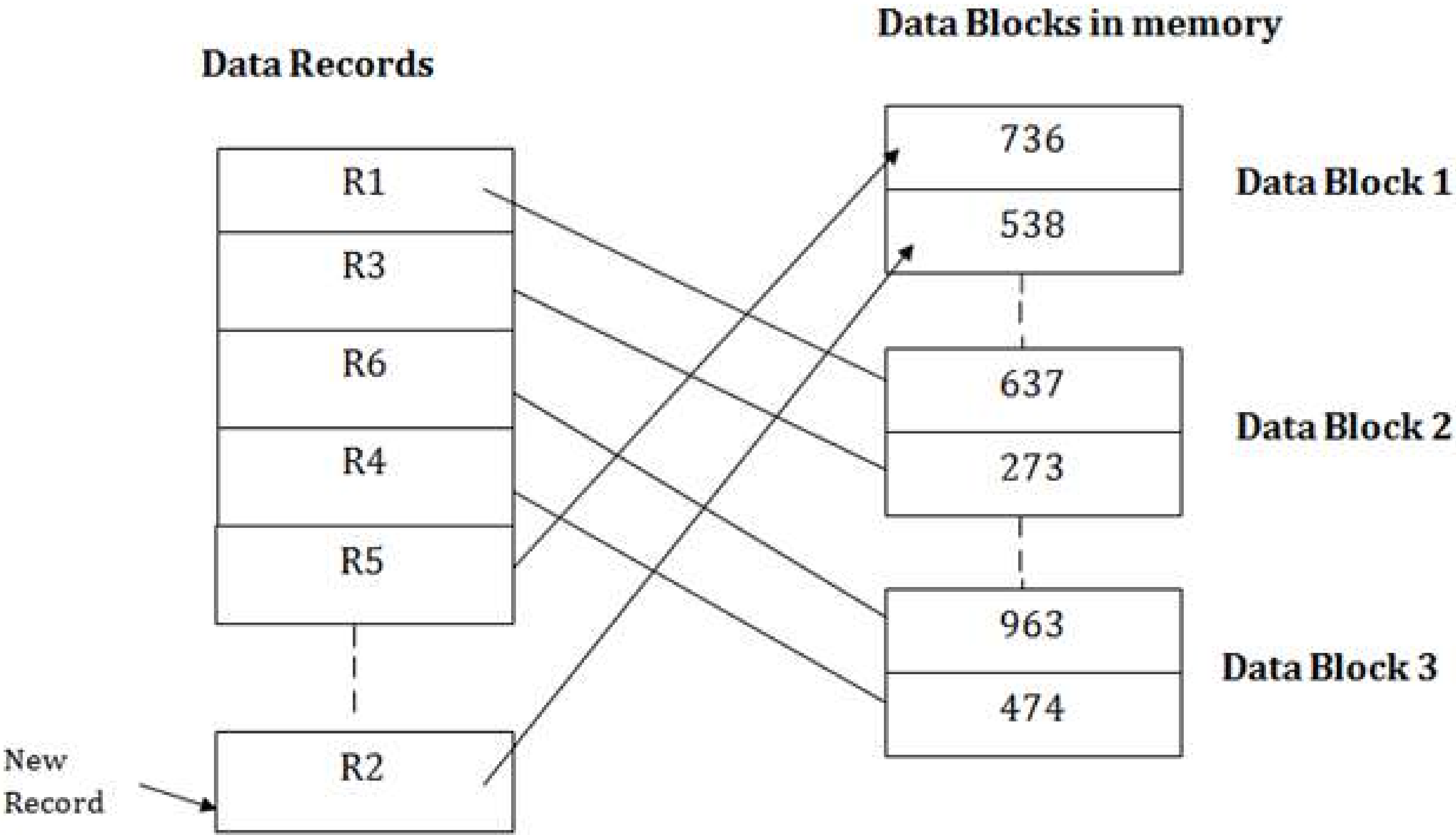
2. Heap file organization

- It is the simplest and most basic type of organization. It works with data blocks.
- In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
- When the data block is full, the new record is stored in some other block.
- This new data block need not to be the very next data block, but it can select any data block in the memory to store new records. The heap file is also known as an unordered file.
- In the file, every record has a unique id, and every page in a file is of the same size. It is the DBMS responsibility to store and manage the new records.
-

Heap file organization(cont..)



Insert a new record



Heap file organization(cont..)

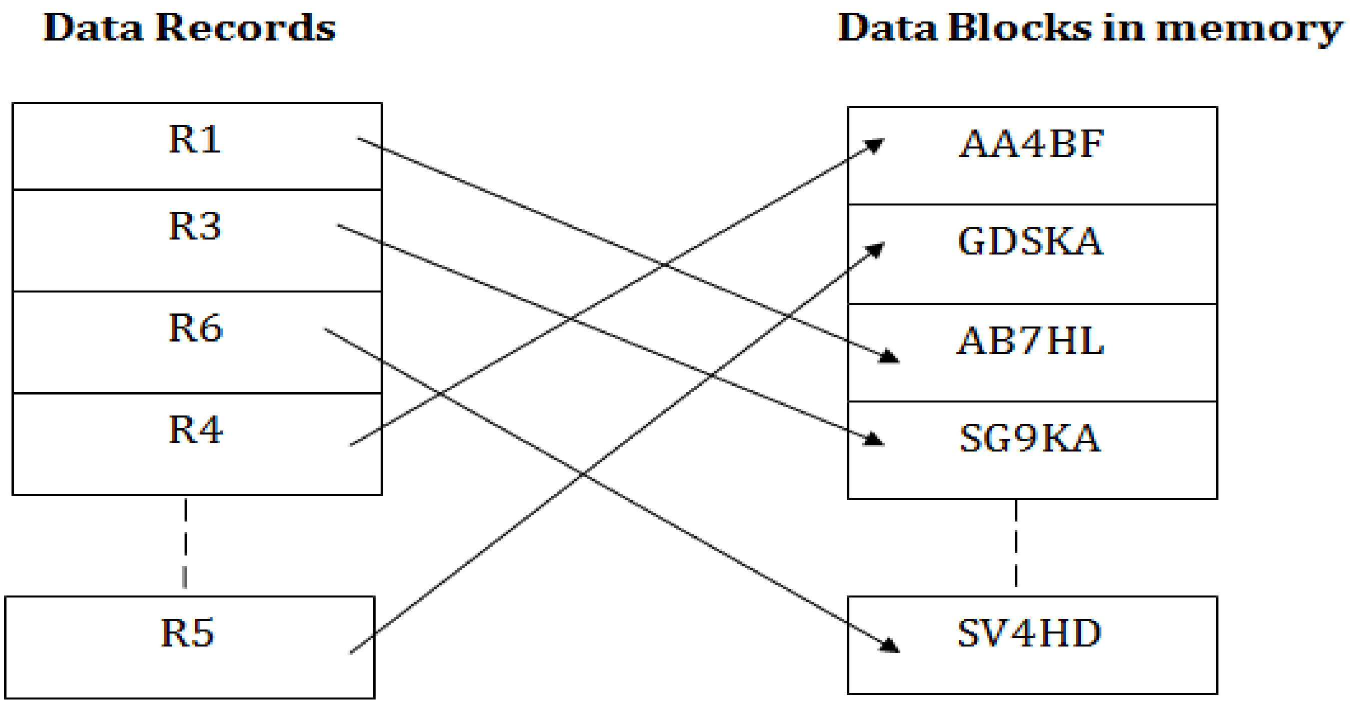
- If we want to search, update or delete the data in heap file organization, then we need to traverse the data from staring of the file till we get the requested record.

- If the database is very large then searching, updating or deleting of record will be time-consuming because there is no sorting or ordering of records.

- In the heap file organization, we need to check all the data until we get the requested record.

3. Hash File Organization

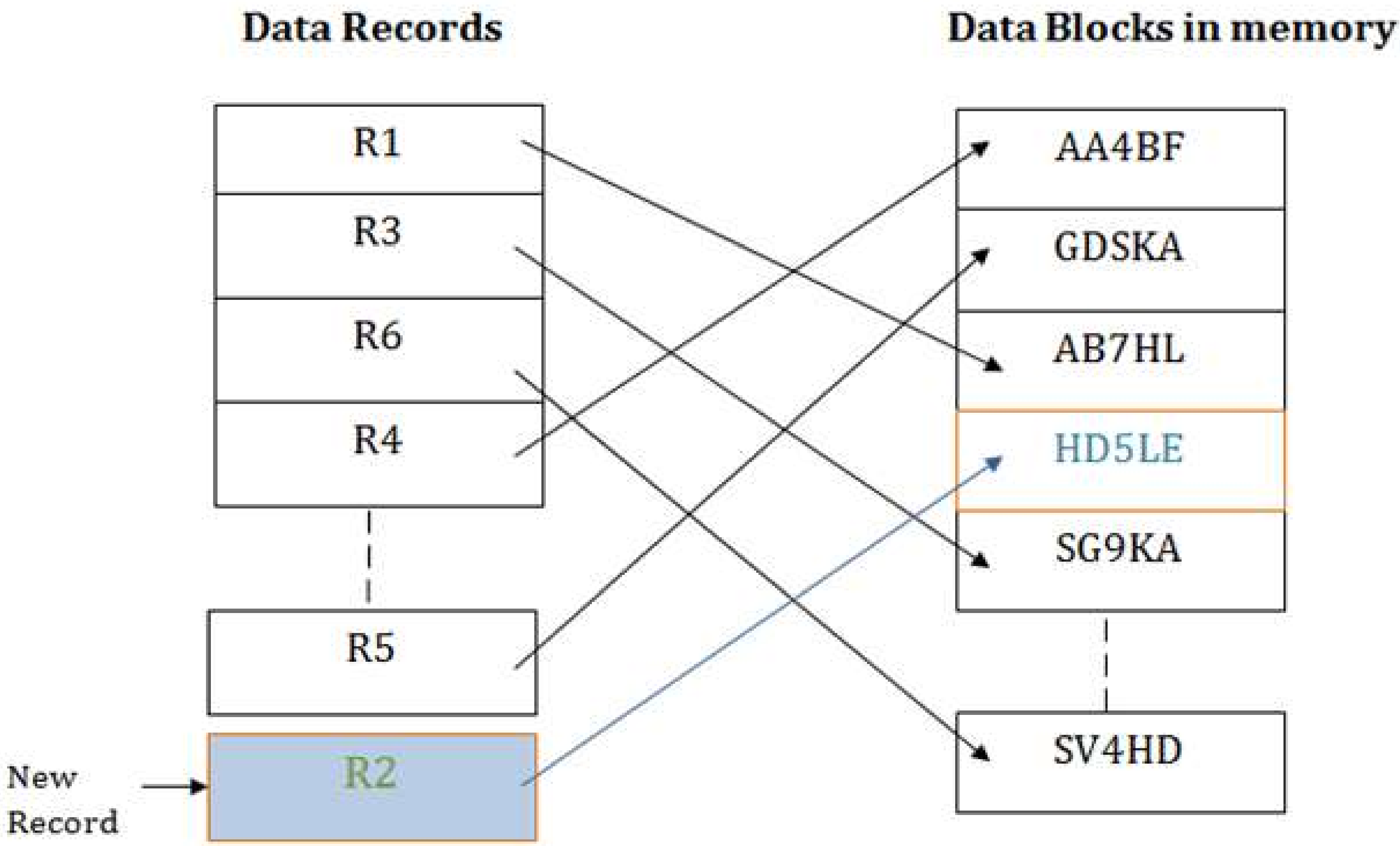
- Hash File Organization uses the computation of hash function on some fields of the records.
- The hash function's output determines the location of disk block where the records are to be placed.



Hash File Organization(cont..)

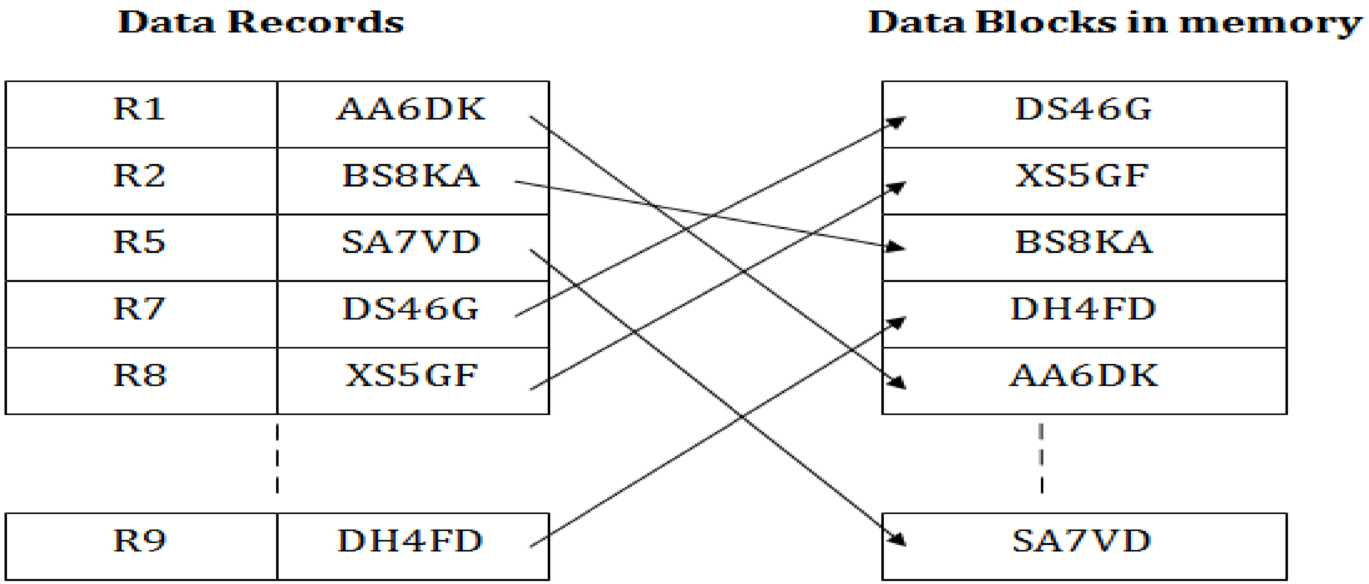
- When a record has to be received using the hash key columns, then the address is generated, and the whole record is retrieved using that address.
- In the same way, when a new record has to be inserted, then the address is generated using the hash key and record is directly inserted.
- The same process is applied in the case of delete and update.
- In this method, there is no effort for searching and sorting the entire file. In this method, each record will be stored randomly in the memory.

Hash File Organization(cont..)



4. Indexed Sequential Access Method (ISAM)

- ISAM method is an advanced sequential file organization.
- In this method, records are stored in the file using the primary key.
- An index value is generated for each primary key and mapped with the record.
- This index contains the address of the record in the file.



Data Records

R1 AA6DK

R2 BS8KA

R5 SA7VD

R7 DS46G

R8 XS5GF

R9 DH4FD

Data Blocks in memory

DS46G

XS5GF

BS8KA

DH4FD

AA6DK

SA7VD

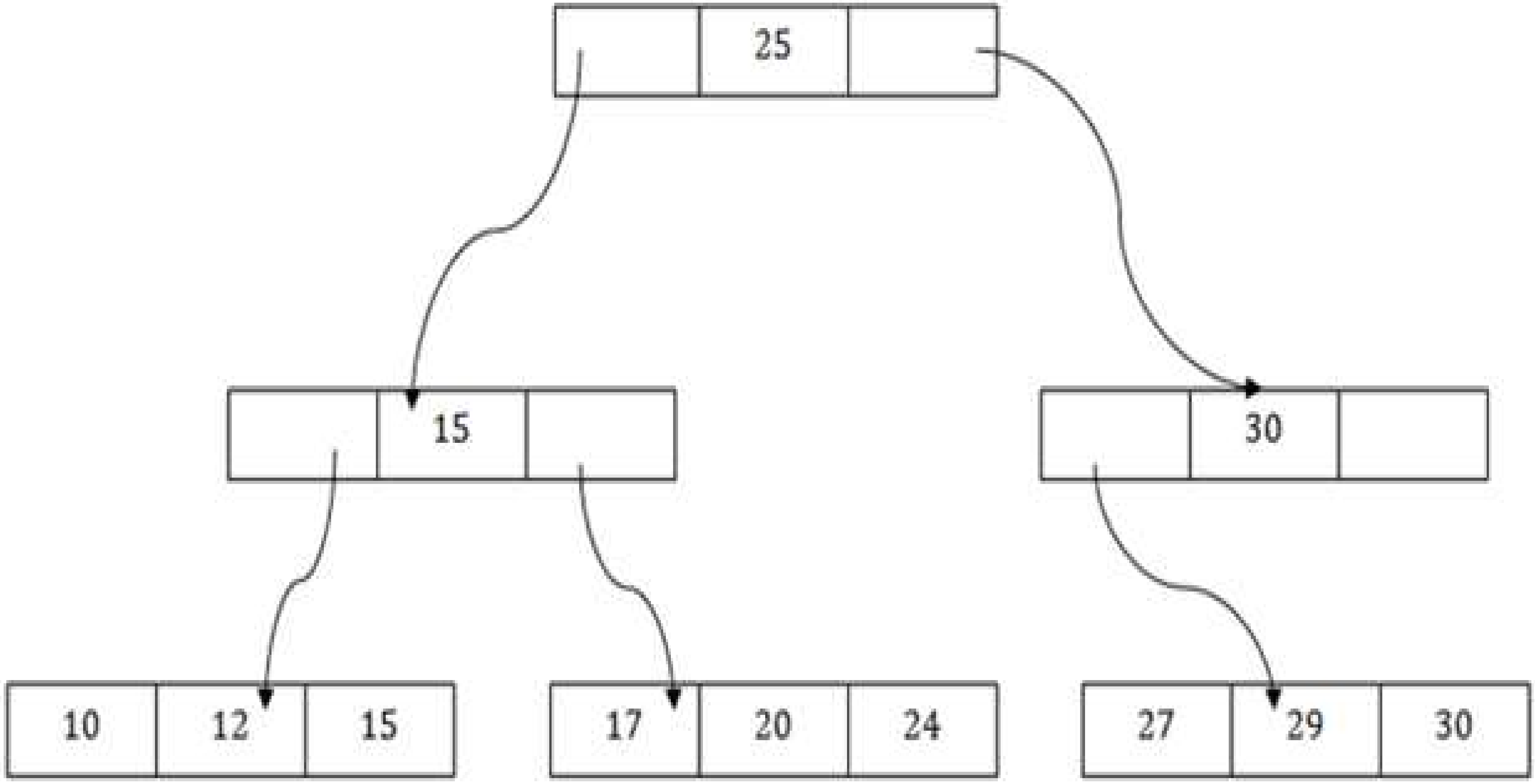
Indexed Sequential Access Method (ISAM)-Cont..

- If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

5.B+ File Organization

- B+ tree file organization is the advanced method of an indexed sequential access method.
- It uses a tree-like structure to store records in File.
- It uses the same concept of key-index where the primary key is used to sort the records.
- For each primary key, the value of the index is generated and mapped with the record.
- The B+ tree is similar to a binary search tree (BST), but it can have more than two children.
- In this method, all the records are stored only at the leaf node.
- Intermediate nodes act as a pointer to the leaf nodes. They do not contain any records.

5.B+ File Organization(Cont..)



The above B+ tree shows that:

- There is one root node of the tree, i.e., 25.

- There is an intermediary layer with nodes. They do not store the actual record. They have only pointers to the leaf node.

- The nodes to the left of the root node contain the prior value of the root and nodes to the right contain next value of the root, i.e., 15 and 30 respectively.

- There is only one leaf node which has only values, i.e., 10, 12, 17, 20, 24, 27 and 29.

- Searching for any record is easier as all the leaf nodes are balanced.

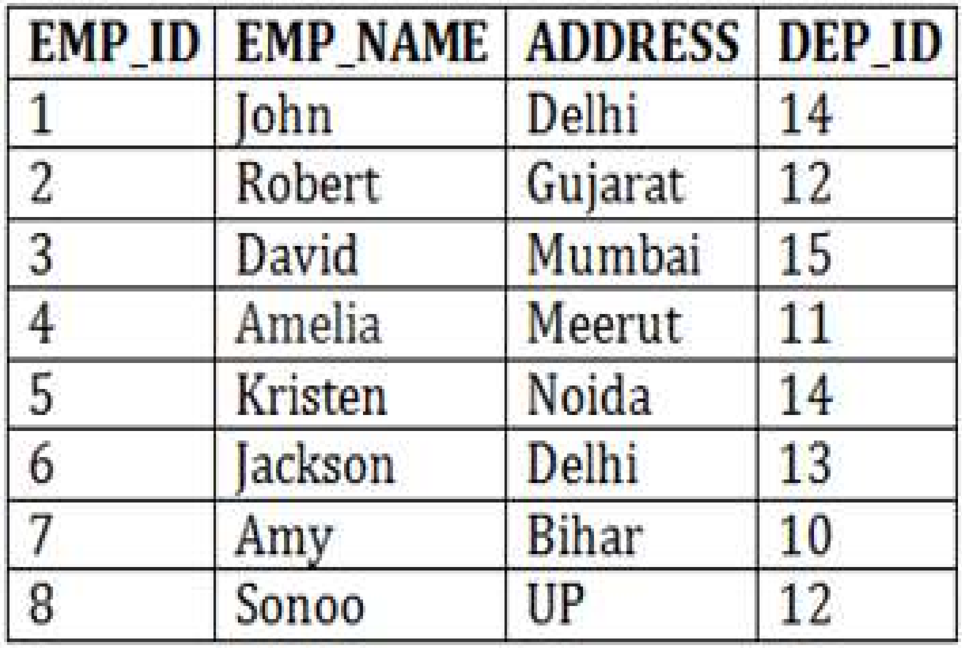
- In this method, searching any record can be traversed through the single path and accessed easily.

6.Cluster File Organization

- When the two or more records are stored in the same file, it is known as clusters.
- These files will have two or more tables in the same data block, and key attributes which are used to map these tables together are stored only once.
- This method reduces the cost of searching for various records in different files.
- The cluster file organization is used when there is a frequent need for joining the tables with the same condition.
- These joins will give only a few records from both tables. In the given example, we are retrieving the record for only particular departments.
- This method can't be used to retrieve the record for the entire department.

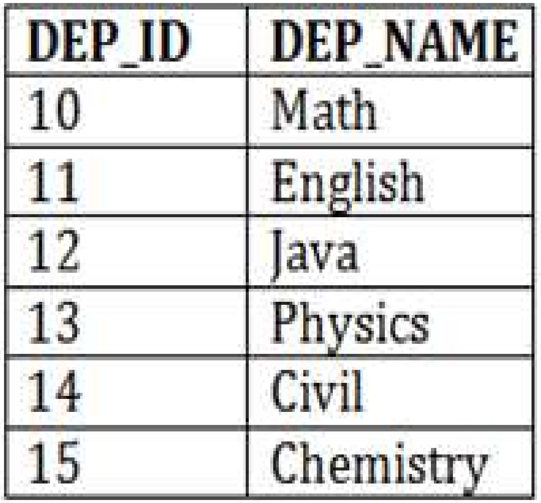
Cluster File Organization(cont..)

EMPLOYEE



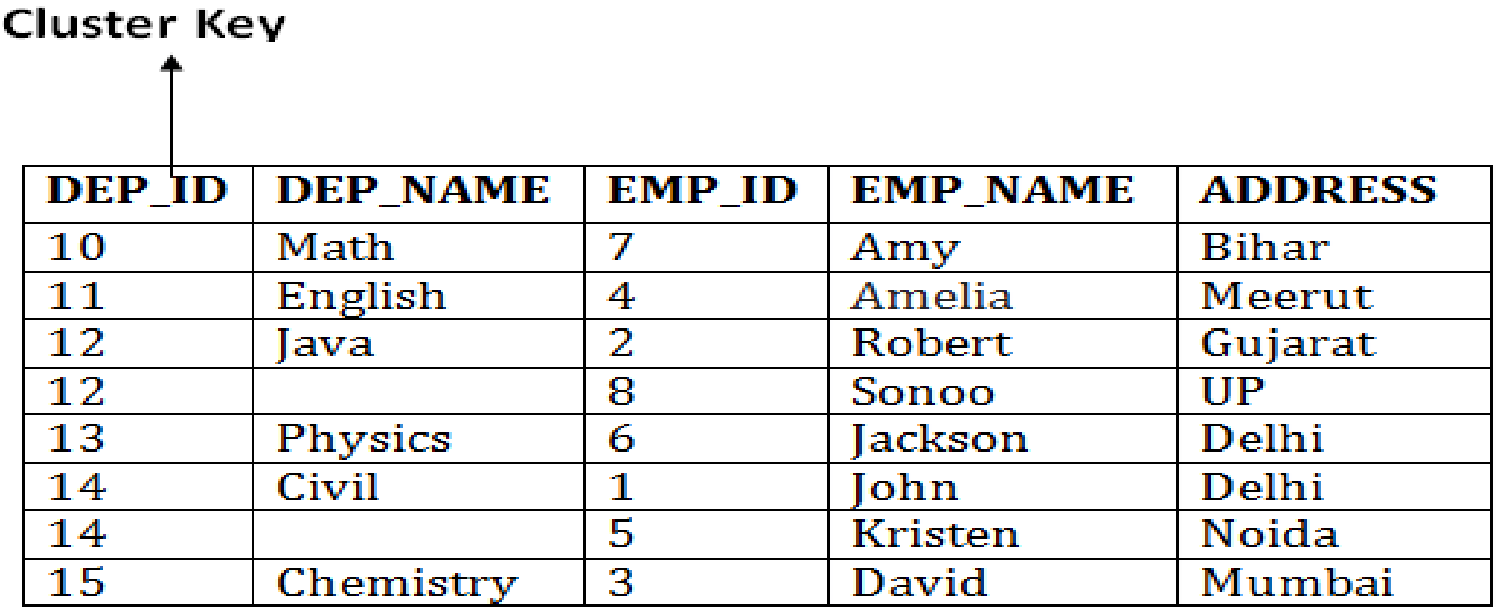
EMP_ID	EMP_NAME	ADDRESS	DEP_ID
1	John	Delhi	14
2	Robert	Gujarat	12
3	David	Mumbai	15
4	Amelia	Meerut	11
5	Kristen	Noida	14
6	Jackson	Delhi	13
7	Amy	Bihar	10
8	Sonoo	UP	12

DEPARTMENT



DEP_ID	DEP_NAME
10	Math
11	English
12	Java
13	Physics
14	Civil
15	Chemistry

Cluster File Organization(cont..)



Cluster Key

DEP_ID	DEP_NAME	EMP_ID	EMP_NAME	ADDRESS
10	Math	7	Amy	Bihar
11	English	4	Amelia	Meerut
12	Java	2	Robert	Gujarat
12		8	Sonoo	UP
13	Physics	6	Jackson	Delhi
14	Civil	1	John	Delhi
14		5	Kristen	Noida
15	Chemistry	3	David	Mumbai

- In this method, we can directly insert, update or delete any record. Data is sorted based on the key with which searching is done.
- Cluster key is a type of key with which joining of the table is performed.

Types of Cluster file organization:

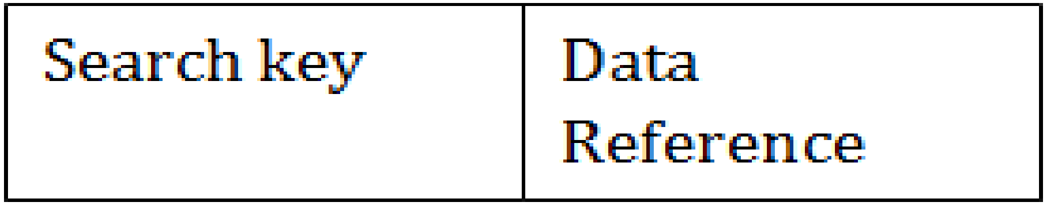
- Cluster file organization is of two types:

- **1. Indexed Clusters:**
- In indexed cluster, records are grouped based on the cluster key and stored together.
- The above EMPLOYEE and DEPARTMENT relationship is an example of an indexed cluster. Here, all the records are grouped based on the cluster key- DEP_ID and all the records are grouped.

- **2. Hash Clusters:**
- It is similar to the indexed cluster. In hash cluster, instead of storing the records based on the cluster key, we generate the value of the hash key for the cluster key and store the records with the same hash key value.

Indexing

- Indexing is used to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed.
- The index is a type of data structure. It is used to locate and access the data in a database table quickly.
- **Index structure:** Indexes can be created using some database columns.



Search key	Data Reference

Fig: Structure of Index

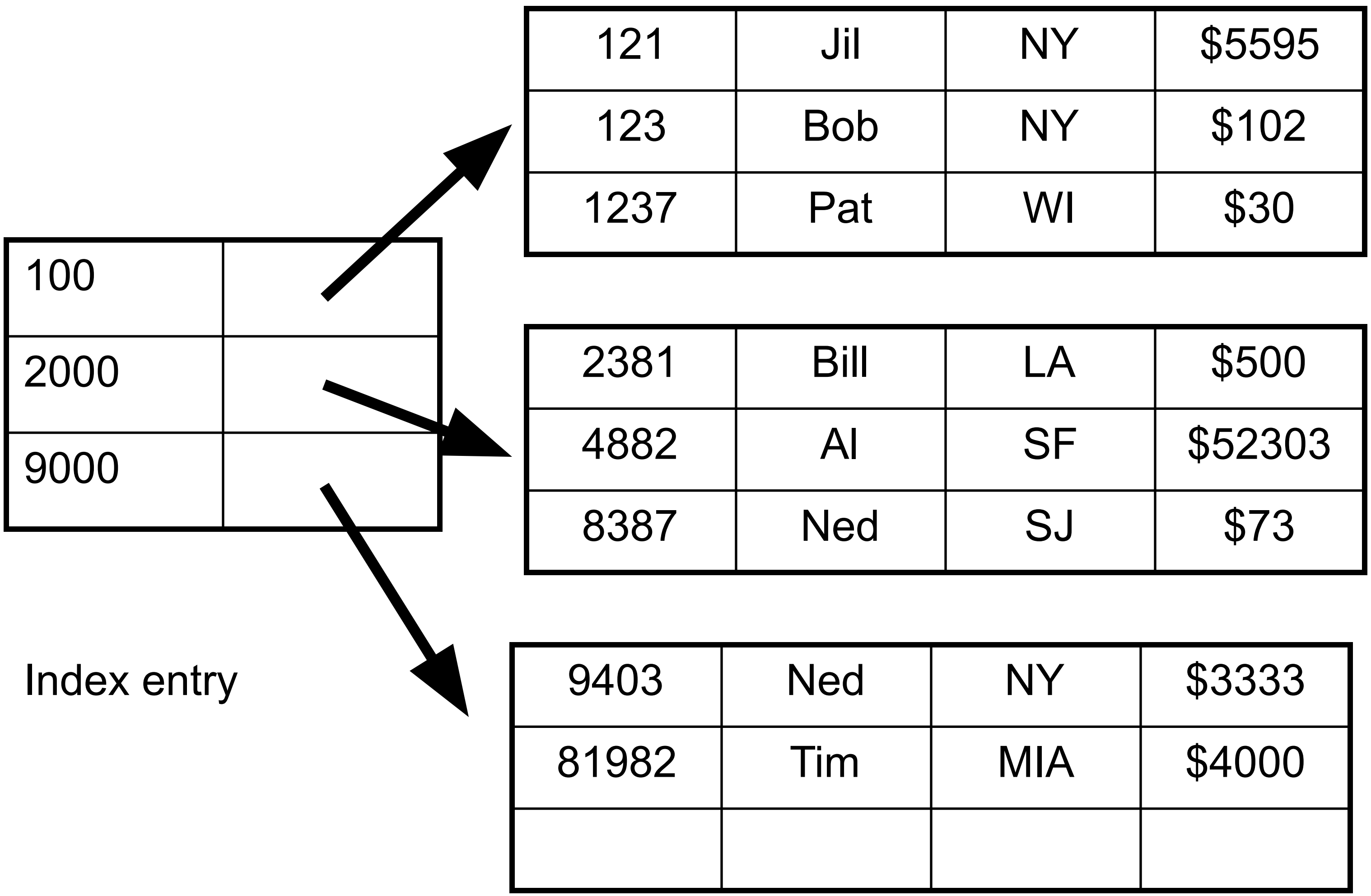
Indexing(Cont..)

- The first column of the database is the search key that contains a copy of the primary key or candidate key of the table.

- The values of the primary key are stored in sorted order so that the corresponding data can be accessed easily.

- The second column of the database is the data reference. It contains a set of pointers holding the address of the disk block where the value of the particular key can be found.

Indexing example



Data entries

121	Jil	NY	$5595
123	Bob	NY	$102
1237	Pat	WI	$30

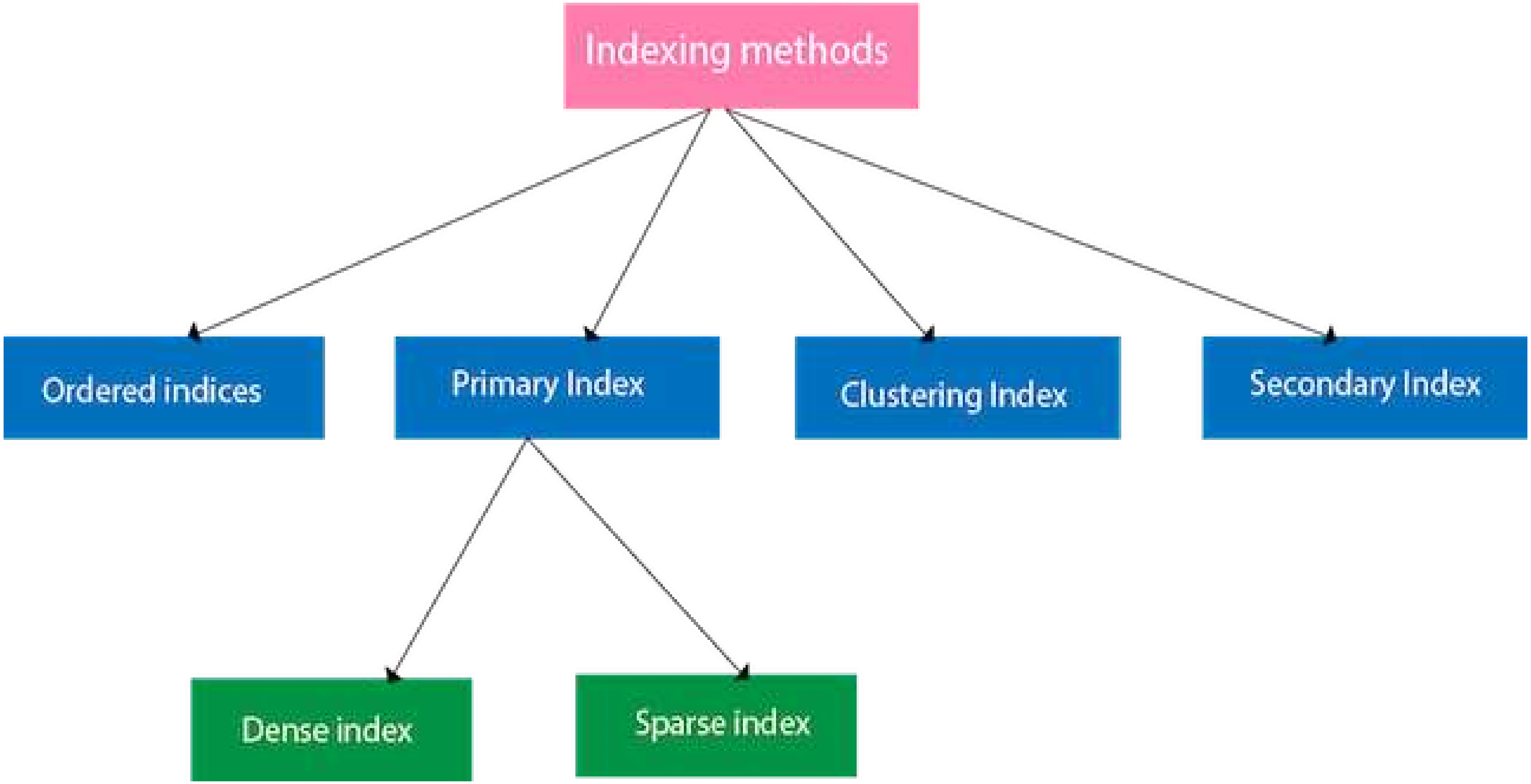
2381	Bill	LA	$500
4882	Al	SF	$52303
8387	Ned	SJ	$73

9403	Ned	NY	$3333
81982	Tim	MIA	$4000

Index entry

100	
2000	
9000	

Indexing Methods



1.Ordered indices

- The indices are usually sorted to make searching faster. The indices which are sorted are known as ordered indices.

- **Example**: Suppose we have an employee table with thousands of record and each of which is 10 bytes long. If their IDs start with 1, 2, 3... .and so on and we have to search student with ID-543.

- In the case of a database with no index, we have to search the disk block from starting till it reaches 543. The DBMS will read the record after reading 543*10=5430 bytes.

- In the case of an index, we will search using indexes and the DBMS will read the record after reading 542*2= 1084 bytes which are very less compared to the previous case.

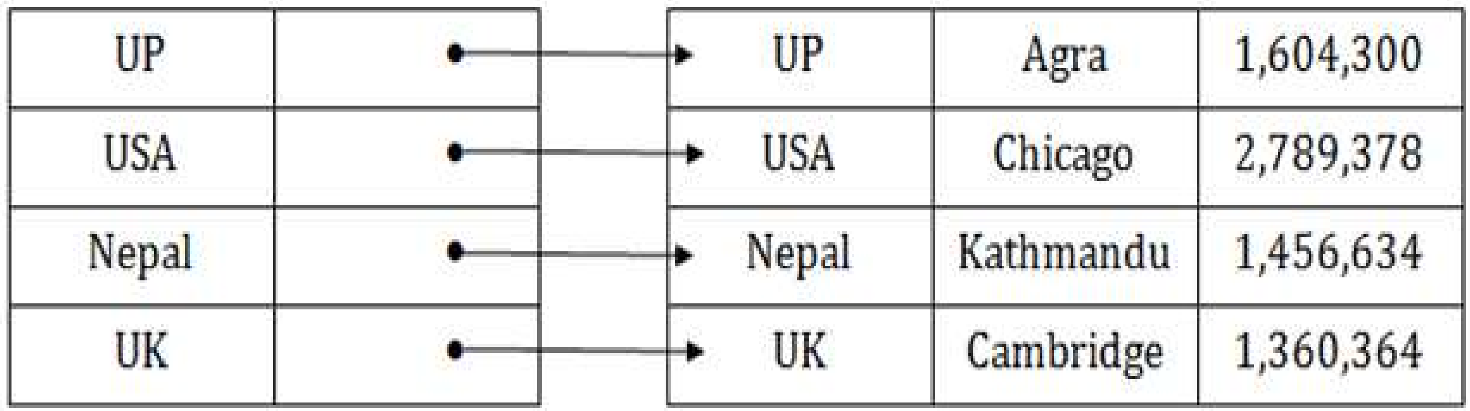
2.Primary Index

- If the index is created on the basis of the primary key of the table, then it is known as primary indexing.
- These primary keys are unique to each record and contain 1:1 relation between the records.
- As primary keys are stored in sorted order, the performance of the searching operation is quite efficient.

- The primary index can be classified into two types:
  a. **Dense index**
  b. **Sparse index**

a. Dense index

- The dense index contains an index record for every search key value in the data file. It makes searching faster.
- In this, the number of records in the index table is same as the number of records in the main table.
- It needs more space to store index record itself. The index records have the search key and a pointer to the actual record on the disk.

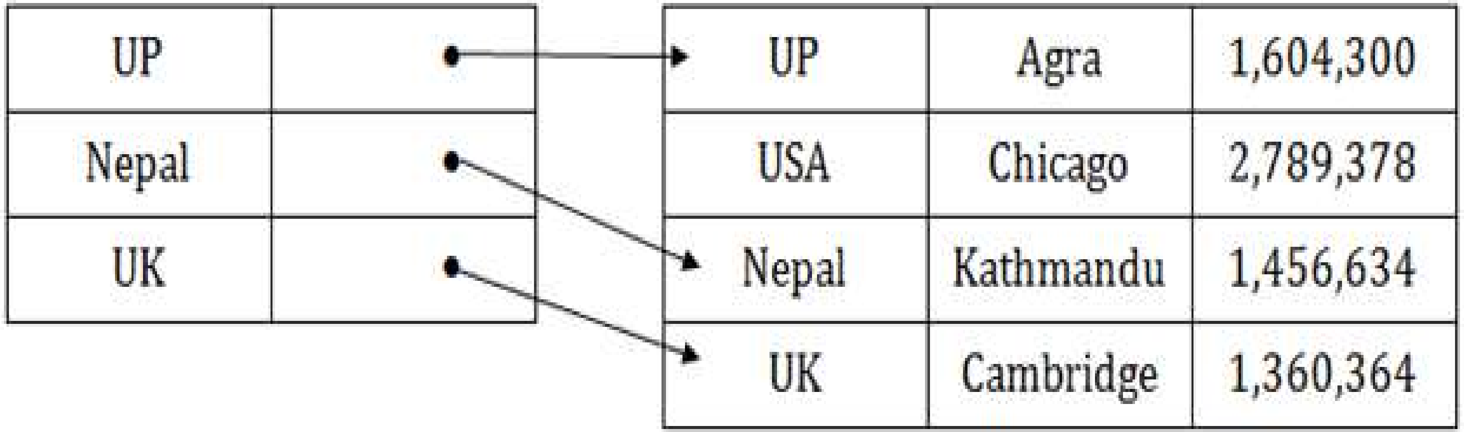


UP	●
USA	●
Nepal	●
UK	●

UP	Agra	1,604,300
USA	Chicago	2,789,378
Nepal	Kathmandu	1,456,634
UK	Cambridge	1,360,364

b.Sparse index

- In the data file, index record appears only for a few items. Each item points to a block.
- In this, instead of pointing to each record in the main table, the index points to the records in the main table in a gap.



UP	●
Nepal	●
UK	●

UP	Agra	1,604,300
USA	Chicago	2,789,378
Nepal	Kathmandu	1,456,634
UK	Cambridge	1,360,364

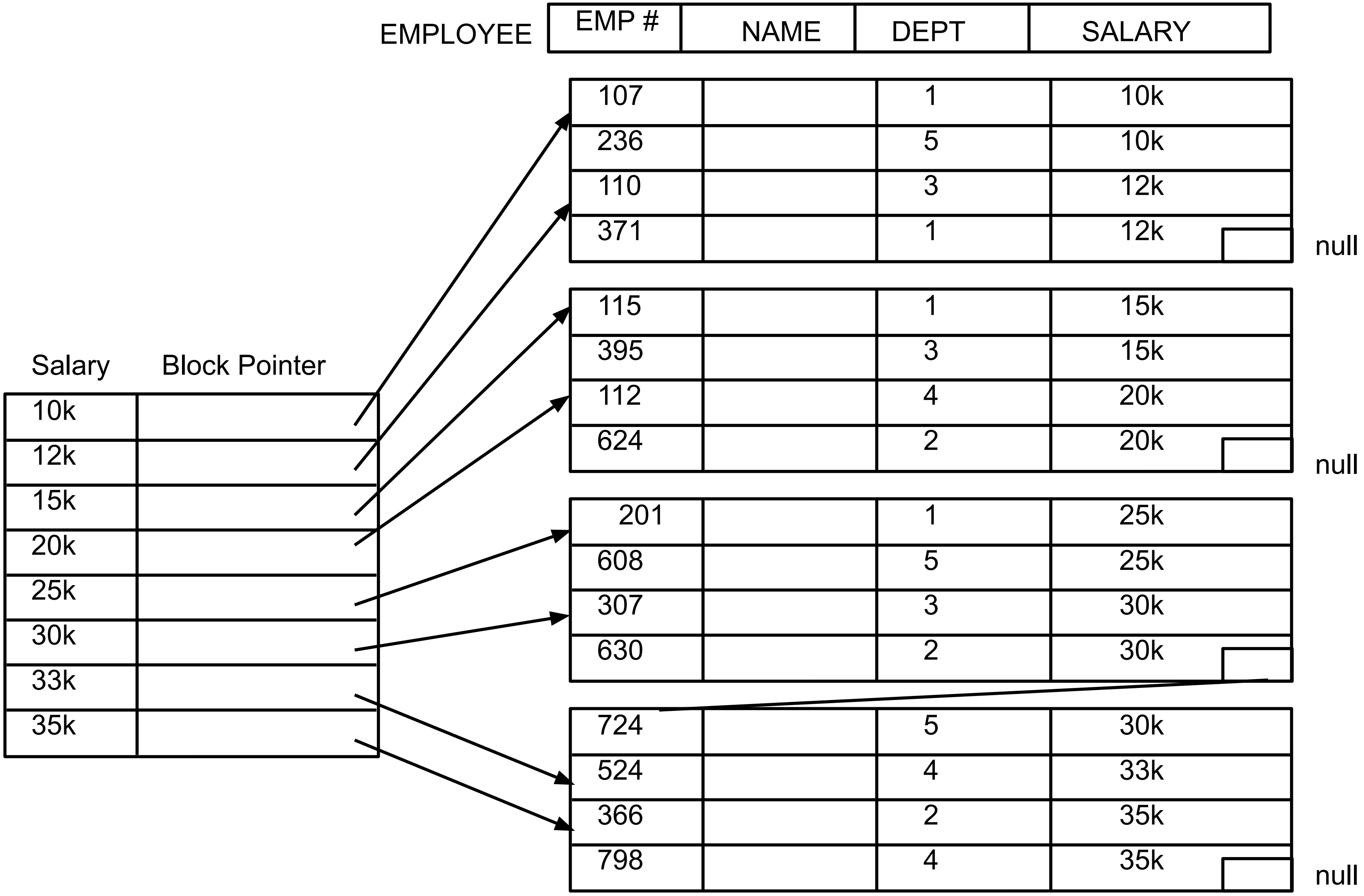
3.Clustering Index

- A clustered index can be defined as an ordered data file. Sometimes the index is created on non-primary key columns which may not be unique for each record.

- In this case, to identify the record faster, we will group two or more columns to get the unique value and create index out of them. This method is called a clustering index.

- The records which have similar characteristics are grouped, and indexes are created for these group.

Clustering Index Example....



Salary	Block Pointer
10k	
12k	
15k	
20k	
25k	
30k	
33k	
35k	

EMPLOYEE

EMP #	NAME	DEPT	SALARY
107		1	10k
236		5	10k
110		3	12k
371		1	12k

EMP #	NAME	DEPT	SALARY
115		1	15k
395		3	15k
112		4	20k
624		2	20k

EMP #	NAME	DEPT	SALARY
201		1	25k
608		5	25k
307		3	30k
630		2	30k

EMP #	NAME	DEPT	SALARY
724		5	30k
524		4	33k
366		2	35k
798		4	35k

4.Secondary indexing

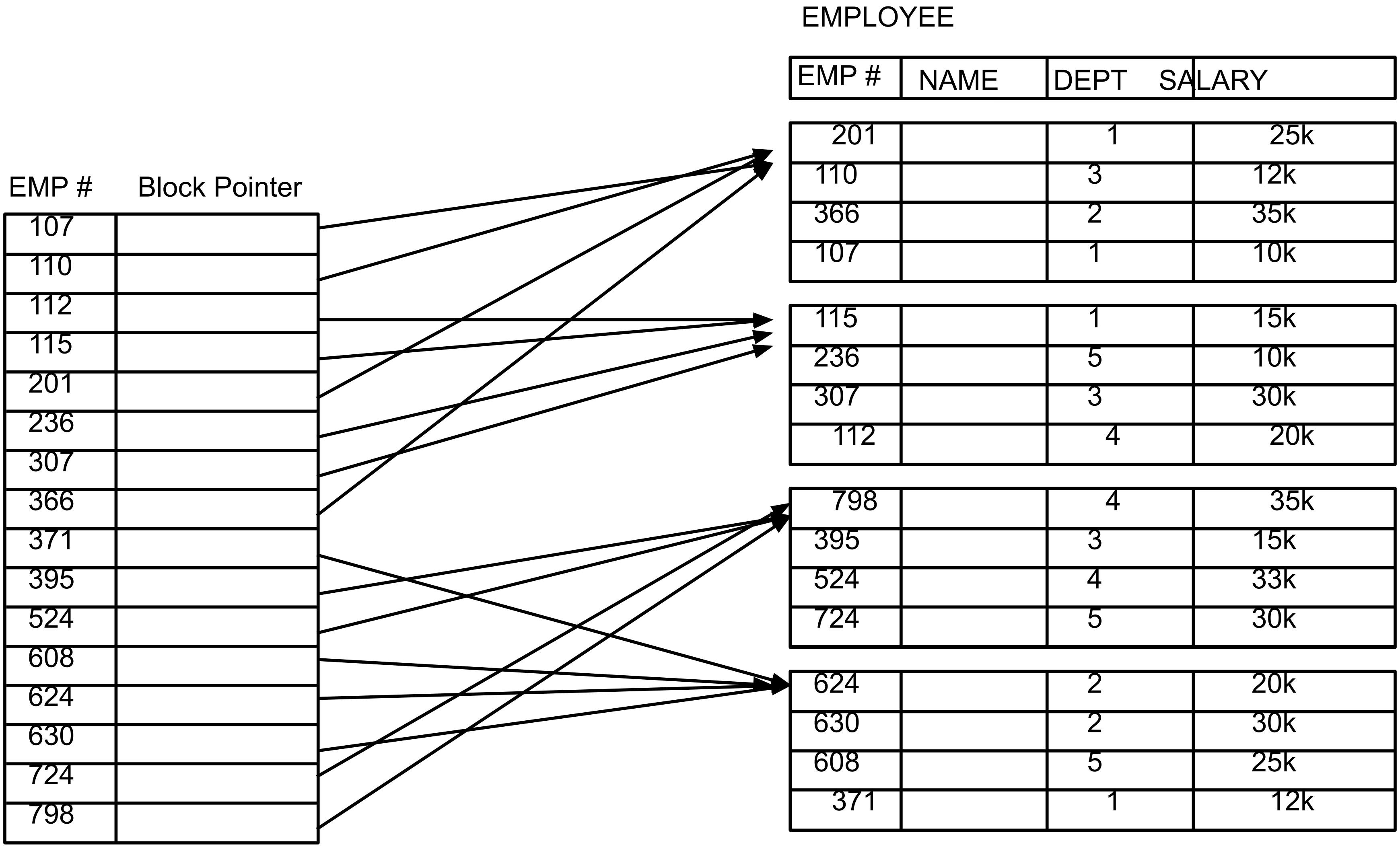
- In secondary indexing, to reduce the size of mapping, another level of indexing is introduced.

- In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small.

- Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster.

- The mapping of the second level and actual data are stored in the secondary memory (hard disk).

Secondary indexing Example

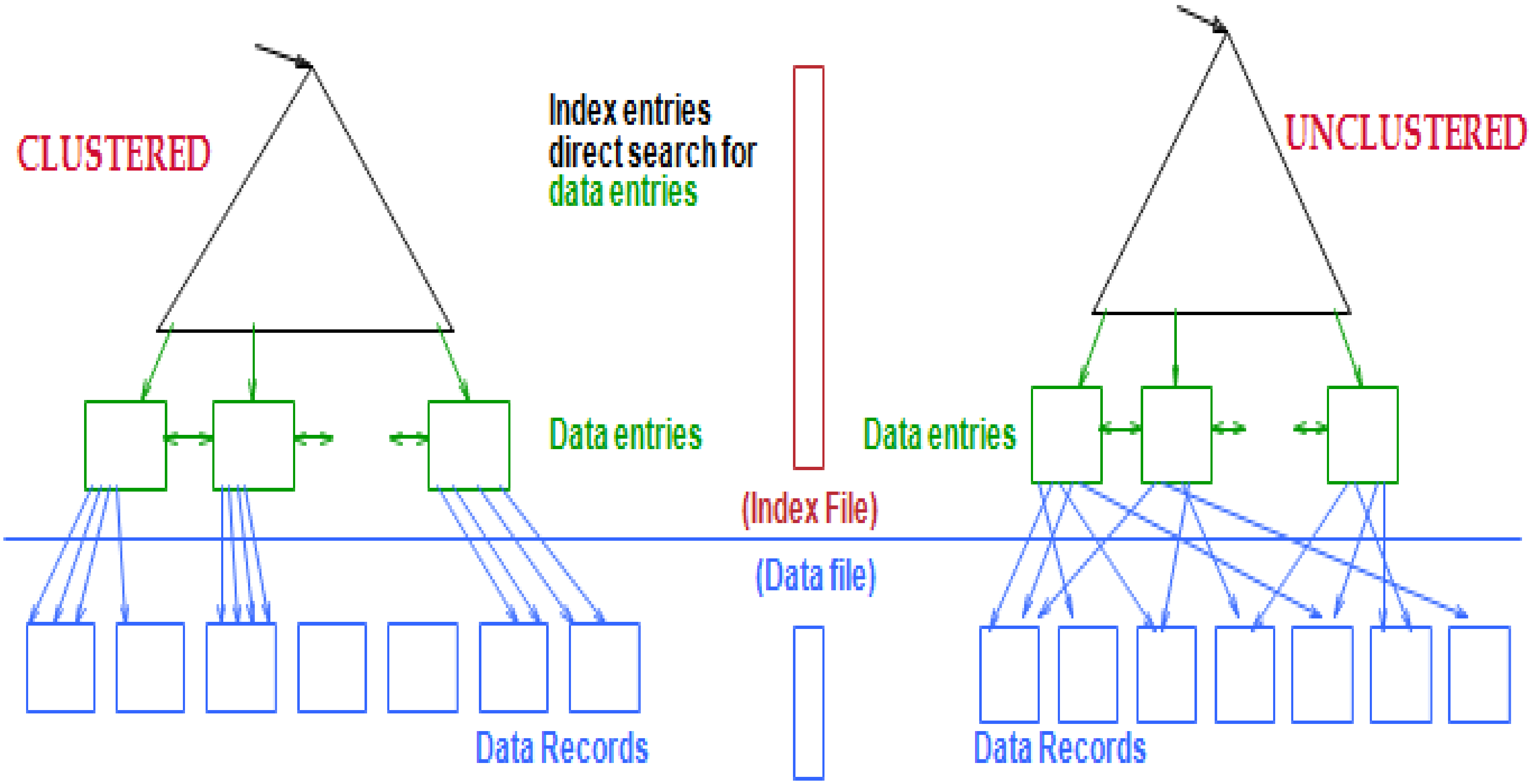


EMP #	Block Pointer
107	
110	
112	
115	
201	
236	
307	
366	
371	
395	
524	
608	
624	
630	
724	
798	

EMPLOYEE

EMP #	NAME	DEPT	SALARY
201		1	25k
110		3	12k
366		2	35k
107		1	10k
115		1	15k
236		5	10k
307		3	30k
112		4	20k
798		4	35k
395		3	15k
524		4	33k
724		5	30k
624		2	20k
630		2	30k
608		5	25k
371		1	12k

Clustered Vs Un clustered(Secondary) Indexes



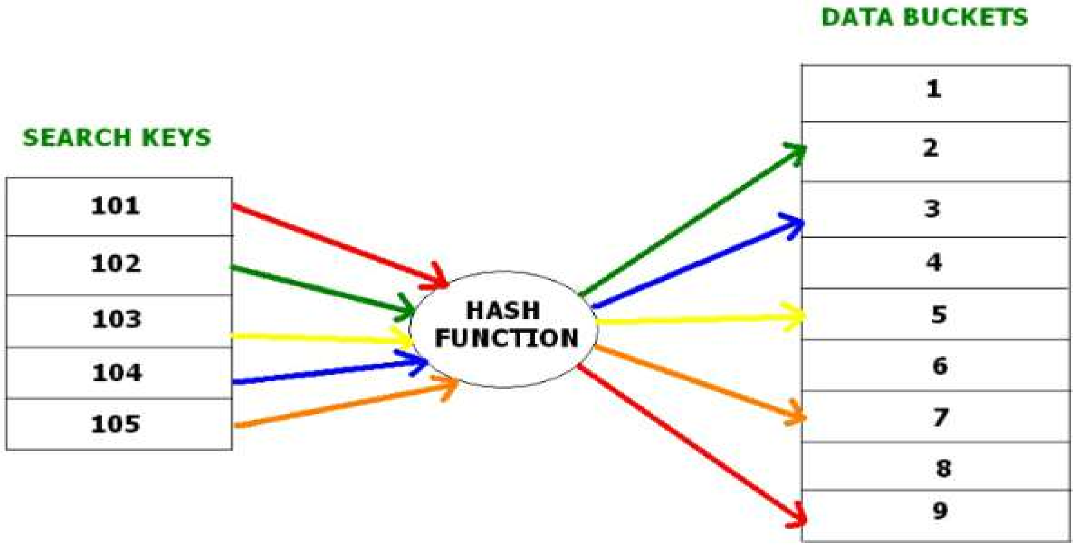
Index Data Structures

- A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure.
- Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed.
- Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records.
- There are 2 types of indexes

    1. Hash based indexes
    2. Tree based indexes

1. Hash-Based indexes

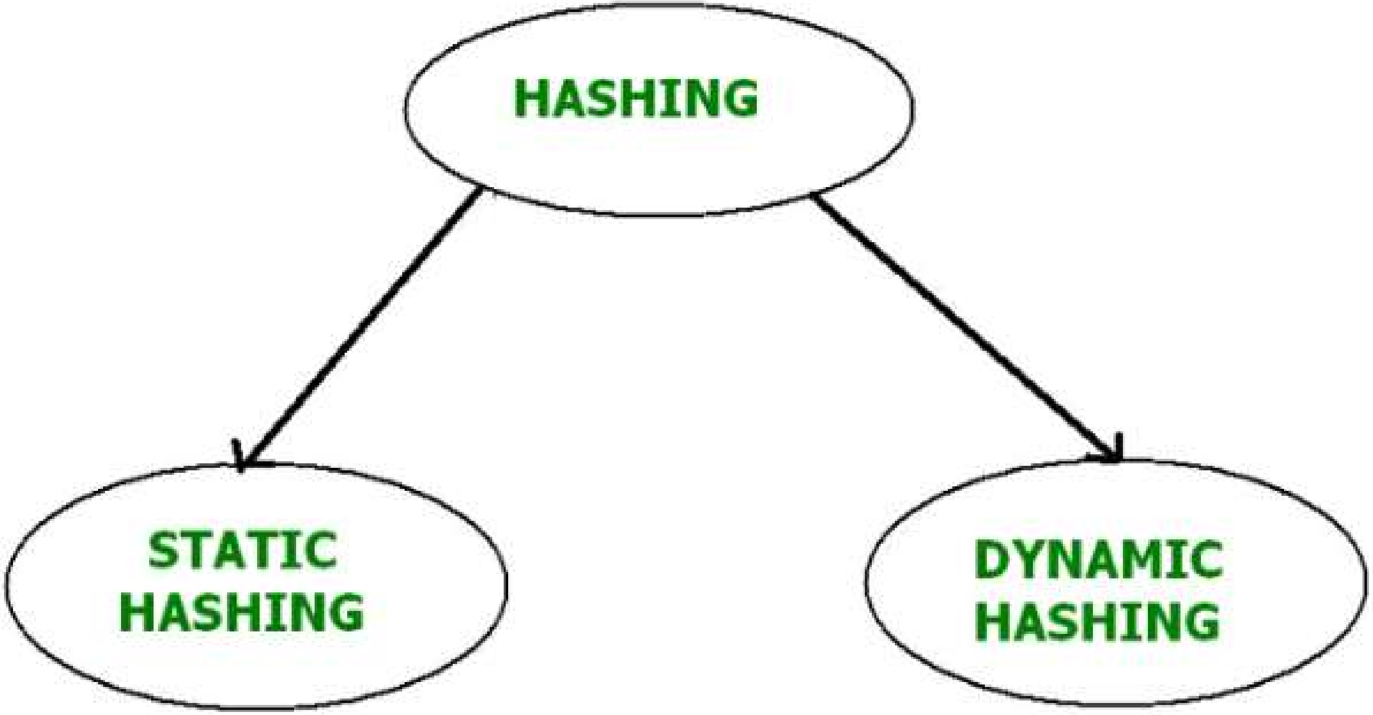
- **In the Hash File Organization we have**
- **Data bucket –** Data buckets are the memory locations where the records are stored. These buckets are also considered as *Unit Of Storage*.
- **Hash Function –** Hash function is a mapping function that maps all the set of search keys to actual record address.
- **Hash Index-**The prefix of an entire hash value is taken as a hash index.



1. Hash-Based indexes

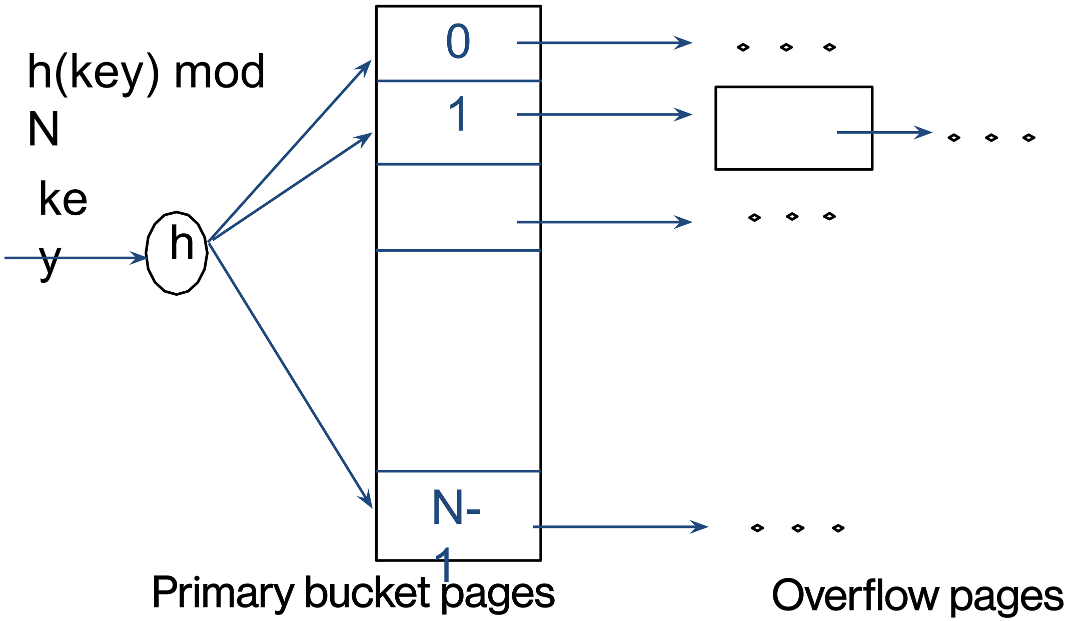
- Index is a collection of **buckets.**
  - Bucket = primary page plus zero or more overflow pages.
  - Buckets contain data entries.
- **Hashing function h:**
  - **h**(r) = bucket in which data entry for record r belongs.
  - **h** looks at **search key** fields of r.

- Hashing is further divided into two sub categories :



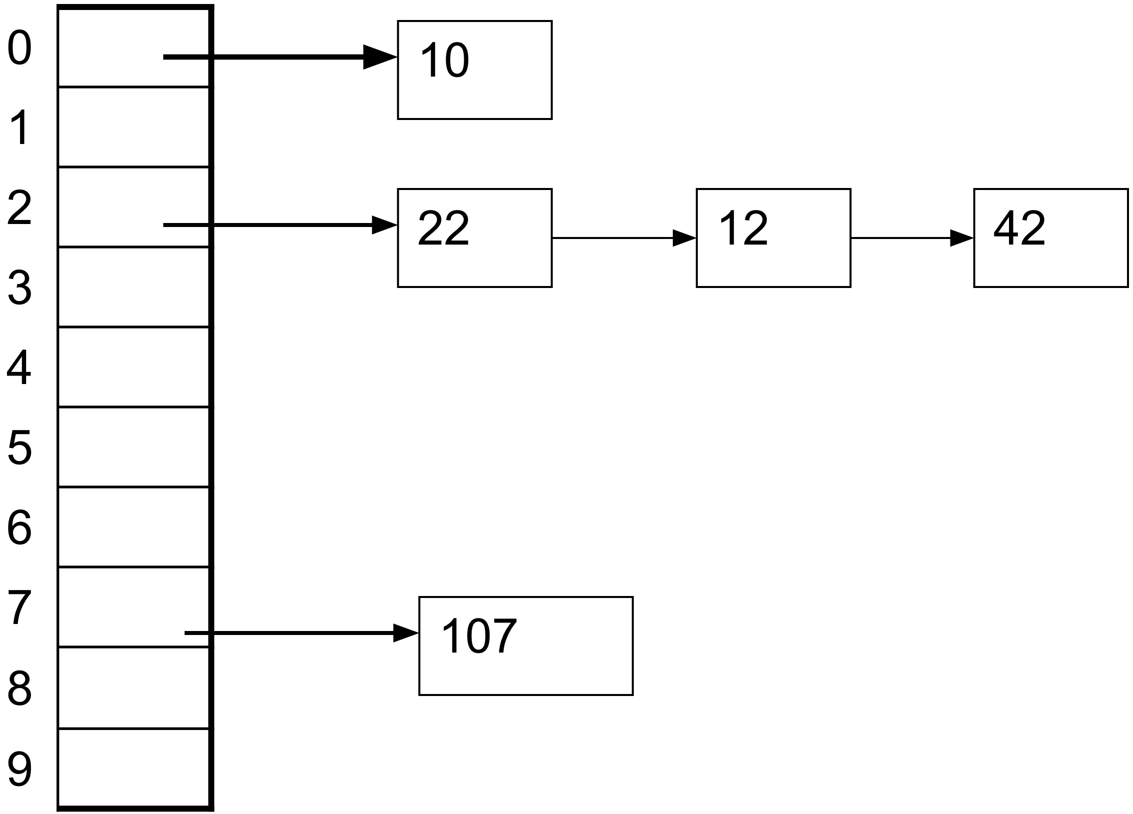
1.Static Hashing

- # primary pages fixed, allocated sequentially, never de-allocated; overflow pages if needed.
- **h(*k*) mod N** = bucket to which data entry with key *k* belongs. (N = # of buckets)



Chaining

- **Chaining**: All keys that map to the same hash value are kept in a linked list

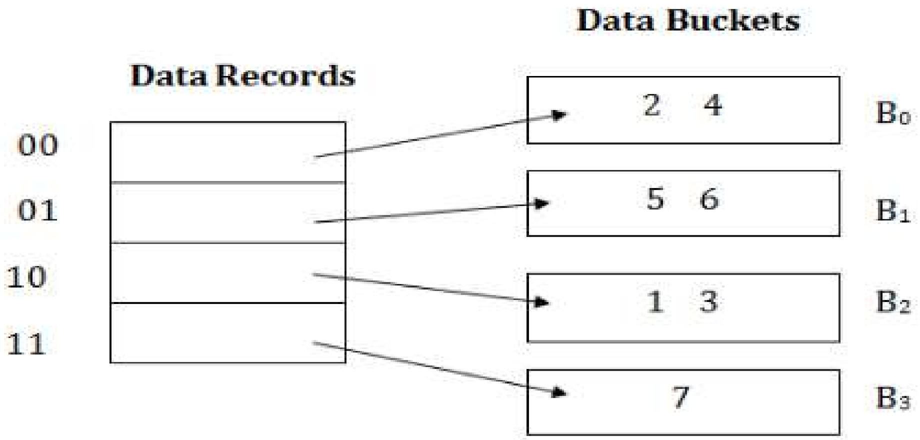


2.Dynamic Hashing

- The dynamic hashing method is used to overcome the problems of static hashing like bucket overflow.

- In this method, data buckets grow or shrink as the records increases or decreases. This method is also known as Extendable hashing method.

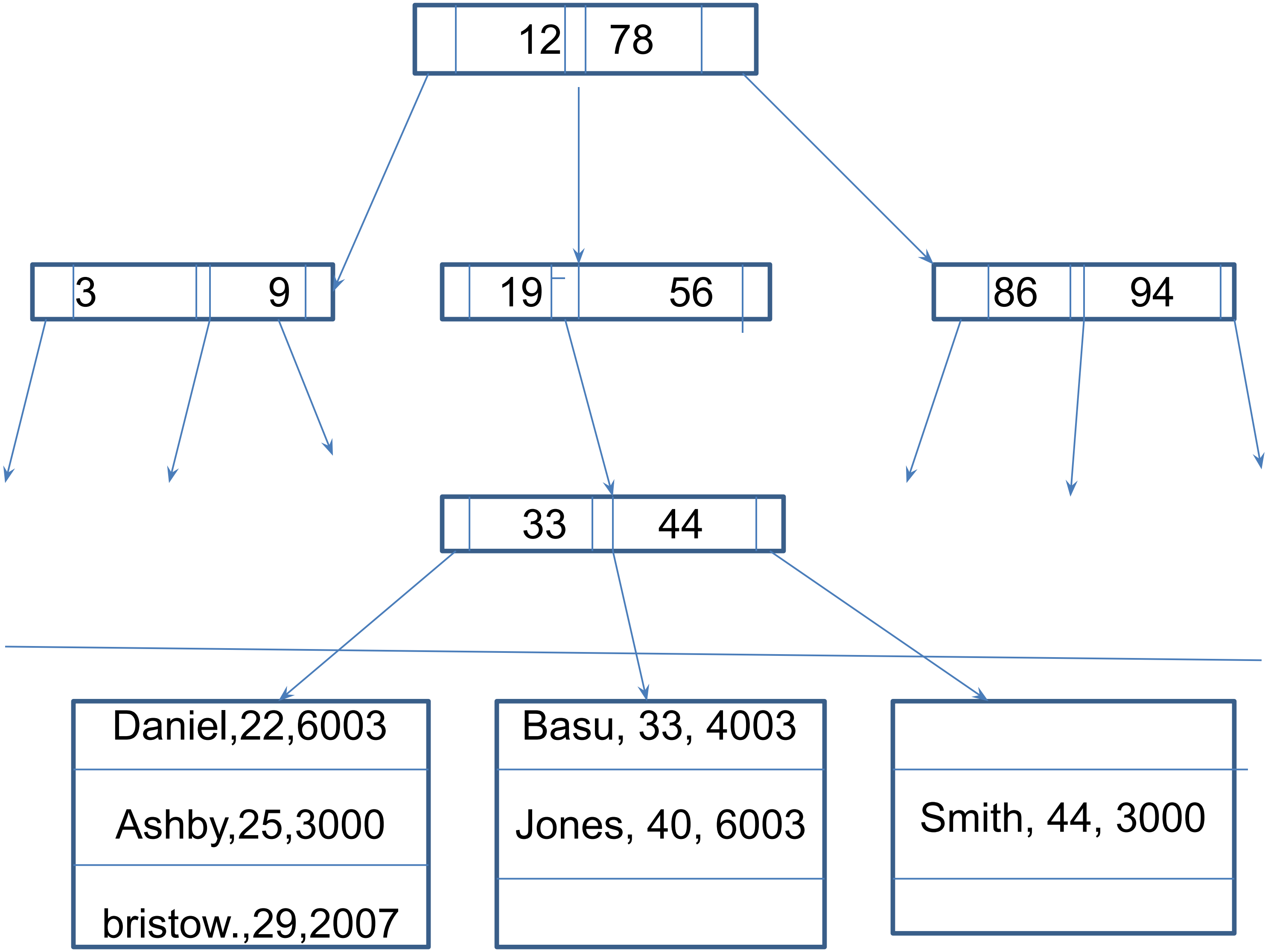
- This method makes hashing dynamic, i.e., it allows insertion or deletion without resulting in poor performance.



2.Tree based Indexing

- An alternative to hash based indexing is to organize a record in a tree like data structure.
- The lowest level of the tree, called the leaf level, contains data entries.
- If we want to add new record, if the value is less than parent node that appears to the left side and if it is record value is greater than parent node then it appears to right side.
- This structure allows us to effectively locate all data entries with search key values in a desire range.
- All searches begin at the top most node, called the root , and the contents of pages in non leaf levels direct search to the correct leaf node.

Tree based Indexing Example...



Leaf level