#File Organization   UNIT-4

- The **File** is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.

- File organization is a logical relationship among various records. This method defines how file records are mapped onto 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.

Types of file organization are as follows:

File Organization
- Sequential FO
- Heap FO
- Hash FO
- ISAM
- B+ Tree FO
- Cluster FO
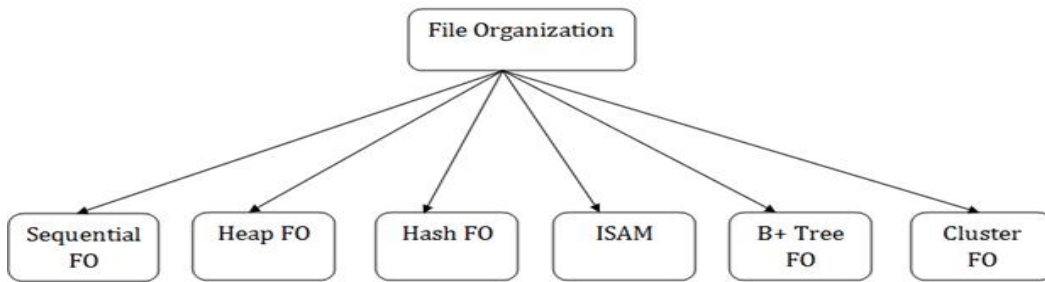
*Sequential File Organization

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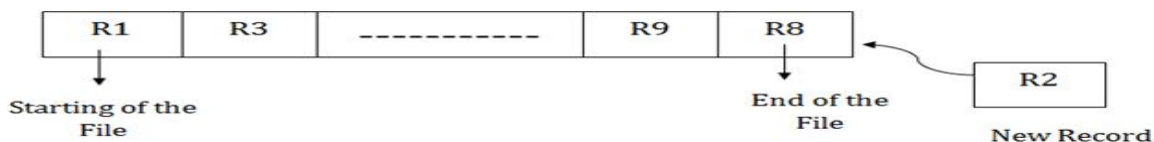
1. Pile File Method:

- 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
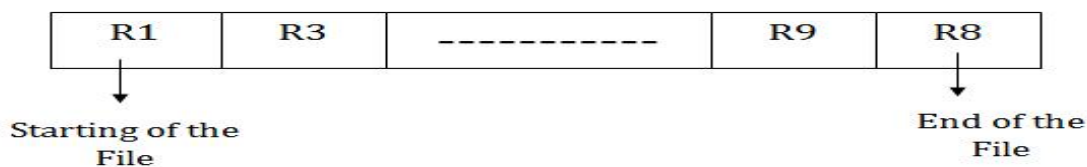
End of the File

Insertion of the new record:

Suppose we have four records R1, R3 and so on upto R9 and R8 in a sequence. Hence, records are nothing but a row in the table. Suppose we want to insert a new record R2 in the sequence, then it will be placed at the end of the file. Here, records are nothing but a row in any table.


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

Starting of the File

End of the File

R2

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.

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

Starting of the File
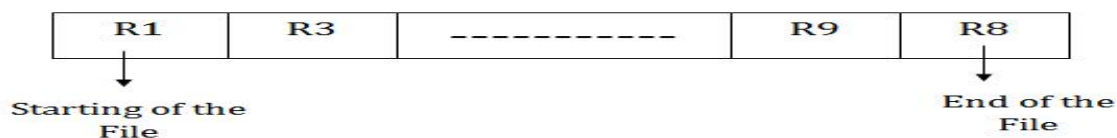
End of the File

Insertion of the new record:

Suppose there is a preexisting sorted sequence of four records R1, R3 and so on upto R6 and R7. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file, and then it will sort the sequence.
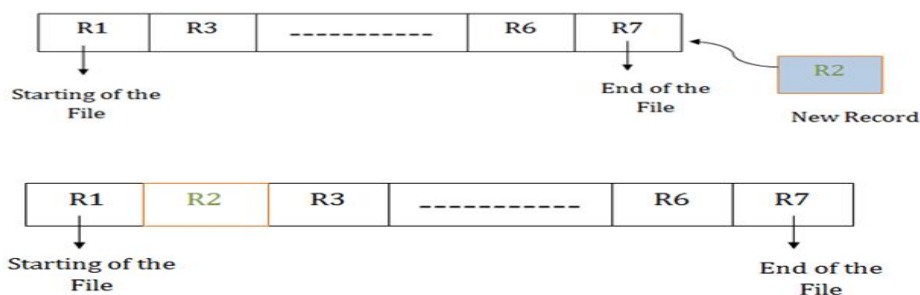
Pros of sequential file organization

- o It contains a fast and efficient method for the huge amount of data.
- o In this method, files can be easily stored in cheaper storage mechanism like magnetic tapes.
- o It is simple in design. It requires no much effort to store the data.
- o This method is used when most of the records have to be accessed like grade calculation of a student, generating the salary slip, etc.
- o This method is used for report generation or statistical calculations.

Cons of sequential file organization

- o It will waste time as we cannot jump on a particular record that is required but we have to move sequentially which takes our time.
- o Sorted file method takes more time and space for sorting the records.

*Heap file organization

- o 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.
- o 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.
- o 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.
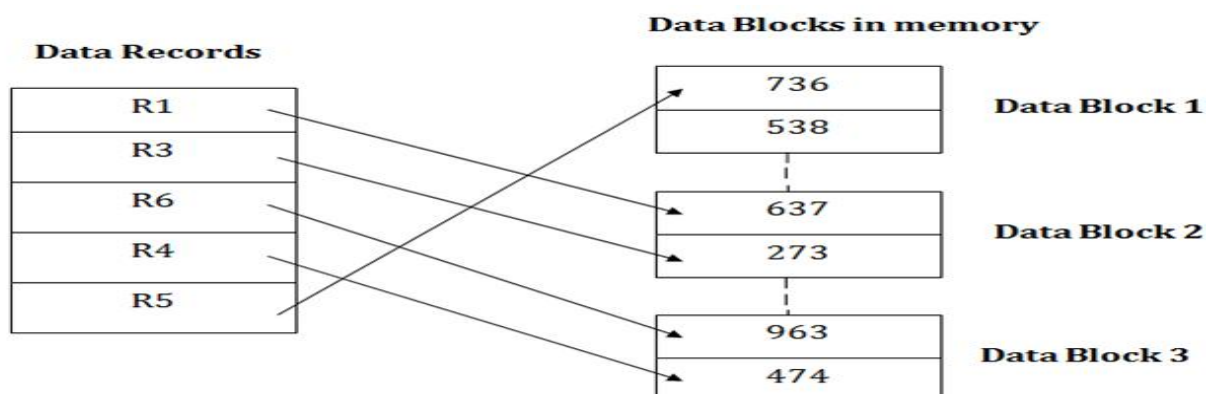
Insertion of a new record

Suppose we have five records R1, R3, R6, R4 and R5 in a heap and suppose we want to insert a new record R2 in a heap. If the data block 3 is full then it will be inserted in any of the database selected by the DBMS, let's say data block 1.

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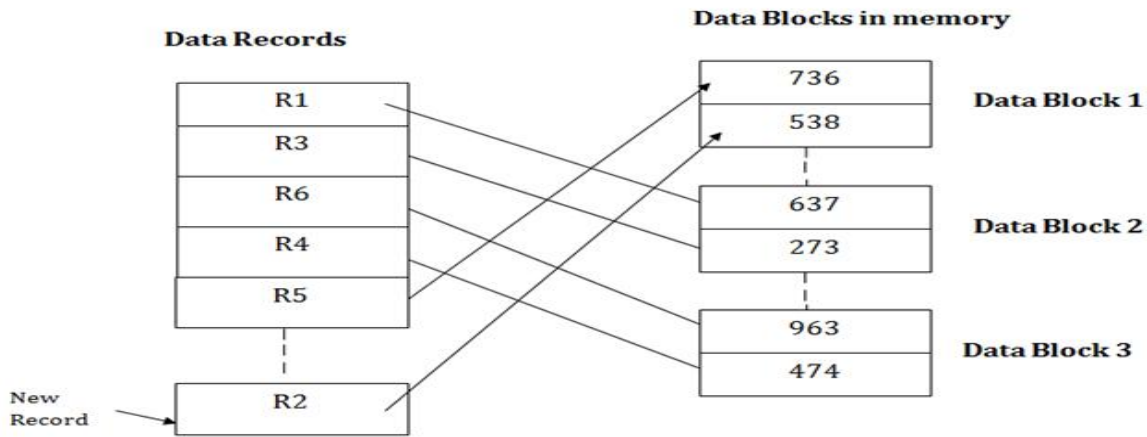
If we want to search, update or delete the data in heap file organization, then we need to traverse the data from staring of the file till we get the requested record.

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.

Pros of Heap file organization

- o It is a very good method of file organization for bulk insertion. If there is a large number of data which needs to load into the database at a time, then this method is best suited.
- o In case of a small database, fetching and retrieving of records is faster than the sequential record.

Cons of Heap file organization

- o This method is inefficient for the large database because it takes time to search or modify the record.
- o This method is inefficient for large databases.

*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.
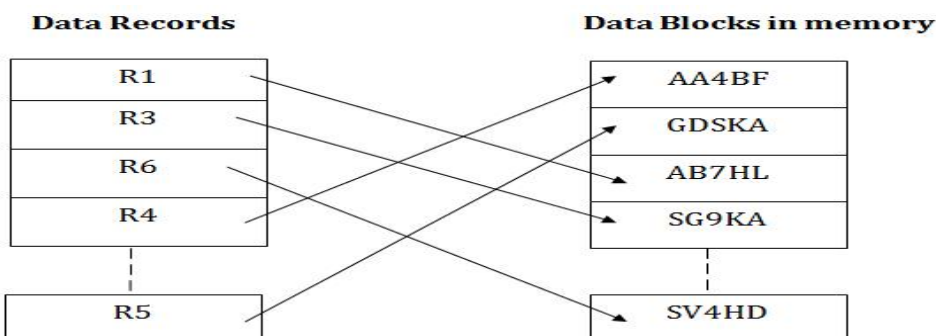

When a record has to be received using the hash key columns, then the address is generated, and the whole record is retrieved using that address. In the same way, when a new record has to be inserted, then the address is generated using the hash key and record is directly inserted. The same process is 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.

Data Records | Data Blocks in memory
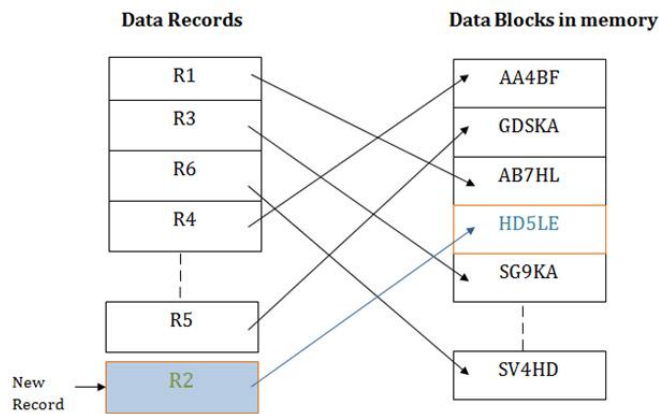
*B+ File Organization

o 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.

o 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.

o 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.
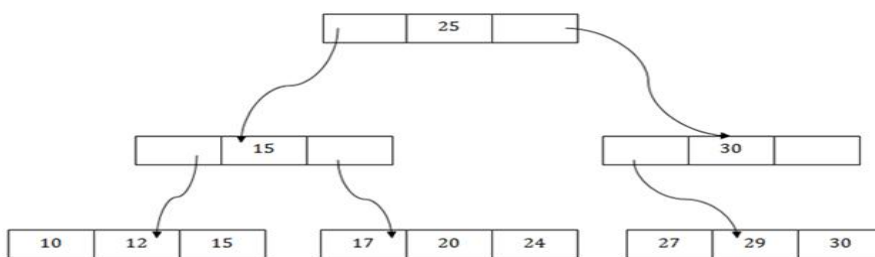
The above B+ tree shows that:

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

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

o 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.

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

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

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

Pros of B+ tree file organization

o In this method, searching becomes very easy as all the records are stored only in the leaf nodes and sorted the sequential linked list.

o Traversing through the tree structure is easier and faster.

o The size of the B+ tree has no restrictions, so the number of records can increase or decrease and the B+ tree structure can also grow or shrink.

o It is a balanced tree structure, and any insert/update/delete does not affect the performance of tree.

Cons of B+ tree file organization

- o   This method is inefficient for the static method.

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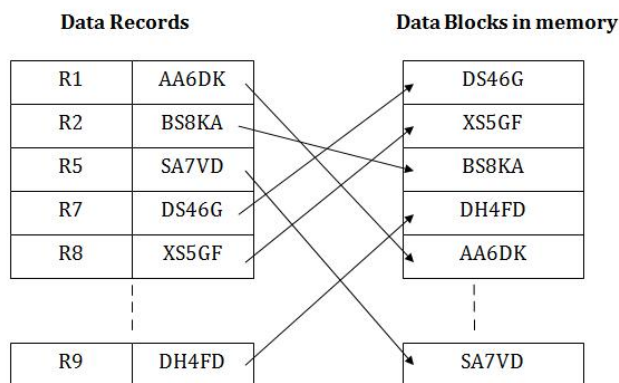
If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

Pros of ISAM:

- o   In this method, each record has the address of its data block, searching a record in a huge database is quick and easy.

- o   This method supports range retrieval and partial retrieval of records. Since the index is based on the primary key values, we can retrieve the data for the given range of value. In the same way, the partial value can also be easily searched, i.e., the student name starting with 'JA' can be easily searched.

Cons of ISAM

- o   This method requires extra space in the disk to store the index value.

- o   When the new records are inserted, then these files have to be reconstructed to maintain the sequence.

- o   When the record is deleted, then the space used by it needs to be released. Otherwise, the performance of the database will slow down.

*Cluster file organization

- o   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.

- o   This method reduces the cost of searching for various records in different files.

- o   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.

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 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.

Pros of Cluster file organization

- o The cluster file organization is used when there is a frequent request for joining the tables with same joining condition.
- o It provides the efficient result when there is a 1:M mapping between the tables.

Cons of Cluster file organization

- o This method has the low performance for the very large database.
- o If there is any change in joining condition, then this method cannot use. If we change the condition of joining then traversing the file takes a lot of time.
- o This method is not suitable for a table with a 1:1 condition.

#Indexing in DBMS

- 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

- 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 Methods
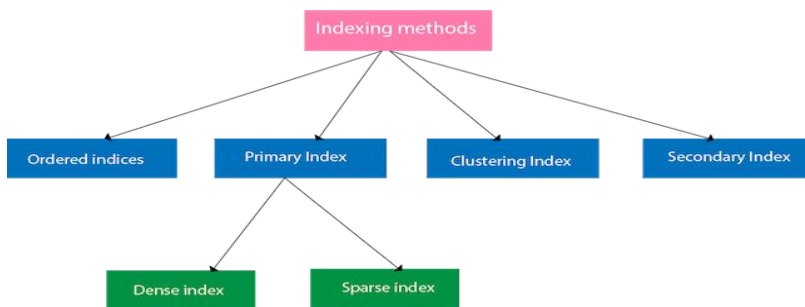
Ordered indices

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.

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: Dense index and Sparse index.

Dense index

- The dense index contains an index record for every search key value in the data file. It makes searching faster.

- o In this, the number of records in the index table is same as the number of records in the main table.
- o 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 | ● | | UP | Agra | 1,604,300 |
USA	●		USA	Chicago	2,789,378
Nepal	●		Nepal	Kathmandu	1,456,634
UK	●		UK	Cambridge	1,360,364

Sparse index

- o In the data file, index record appears only for a few items. Each item points to a block.
- o 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 | ● | | UP | Agra | 1,604,300 |
Nepal	●		USA	Chicago	2,789,378
UK	●		Nepal	Kathmandu	1,456,634
			UK	Cambridge	1,360,364

Clustering Index

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

Example: suppose a company contains several employees in each department. Suppose we use a clustering index, where all employees which belong to the same Dept_ID are considered within a single cluster, and index pointers point to the cluster as a whole. Here Dept_Id is a non-unique key.

Dept_ID	Pointer
1	
2	
3	
4	
5	

1	Record
1	
1	
2	

2	Record
2	
2	
2	

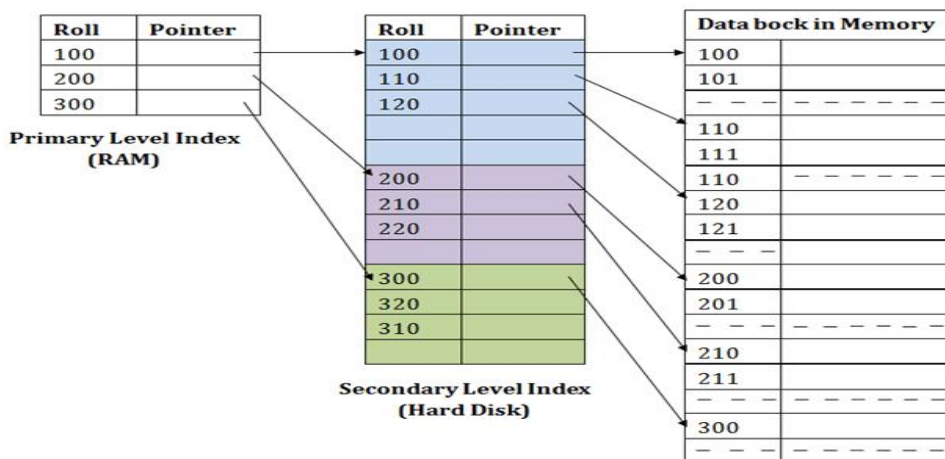| 3 | Record |
3	
3	
4	
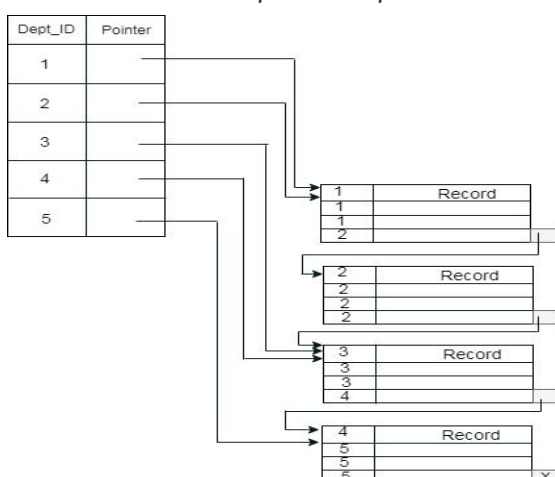
4	Record
5	
5	
5	X

The previous schema is little confusing because one disk block is shared by records which belong to the different cluster. If we use separate disk block for separate clusters, then it is called better technique.
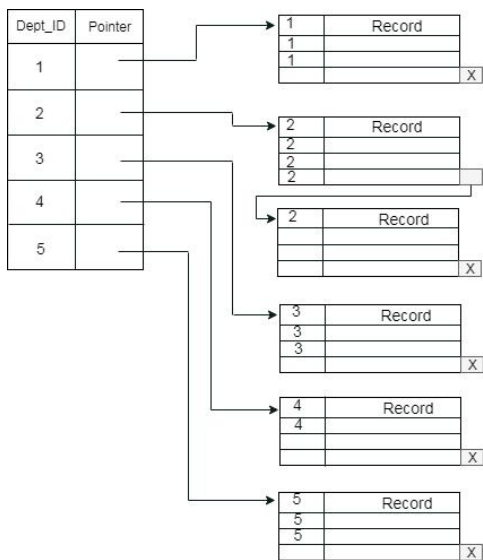
Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

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).

For example:

- o If you want to find the record of roll 111 in the diagram, then it will search the highest entry which is smaller than or equal to 111 in the first level index. It will get 100 at this level.
- o Then in the second index level, again it does max (111) <= 111 and gets 110. Now using the address 110, it goes to the data block and starts searching each record till it gets 111.
- o This is how a search is performed in this method. Inserting, updating or deleting is also done in the same manner.

Bitmap Indexing

It is a special type of indexing built on a single key. But, it is designed to fire queries on multiple keys quickly. We need to arrange the records in sequential order before applying bitmap indexing on it. It makes it simple to fetch a particular record from the block. Also, it becomes easy to allocate them in the block of a file.

Bitmap Index Structure

The word 'bitmap' comprises of 'bit' and 'map'. A bit is the smallest unit of data in a computer system. A map means organizing things. Thus, a bitmap is simply mapping of bits in the form of an array. In a relation, each attribute carries one bitmap for its value. A bitmap has sufficient bits for numbering each record in the block.

For example, consider a relation Student_record where we wish to find out the female and male students whose score in English is greater than 40. The bitmaps for gender are given in the below image.

Applying Bitmap Indexing
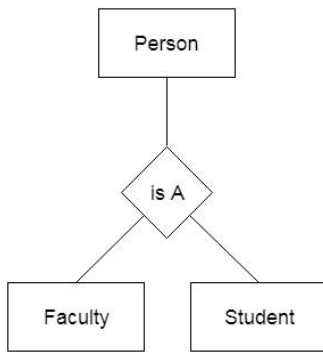on relation 'Student'

#What is Buffer Manager

- o   A Buffer Manager is responsible for allocating space to the buffer in order to store data into the buffer.
- o   If a user request a particular block and the block is available in the buffer, the buffer manager provides the block address in the main memory.
- o   If the block is not available in the buffer, the buffer manager allocates the block in the buffer.
- o   If free space is not available, it throws out some existing blocks from the buffer to allocate the required space for the new block.
- o   The blocks which are thrown are written back to the disk only if they are recently modified when writing on the disk.
- o   If the user requests such thrown-out blocks, the buffer manager reads the requested block from the disk to the buffer and then passes the address of the requested block to the user in the main memory.
- o   However, the internal actions of the buffer manager are not visible to the programs that may create any problem in disk-block requests. The buffer manager is just like a virtual machine.

Generalization

- o   Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- o   In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
- o   Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- o   In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

For example, Faculty and Student entities can be generalized and create a higher level entity Person.

Specialization

- o Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
- o Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- o Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.
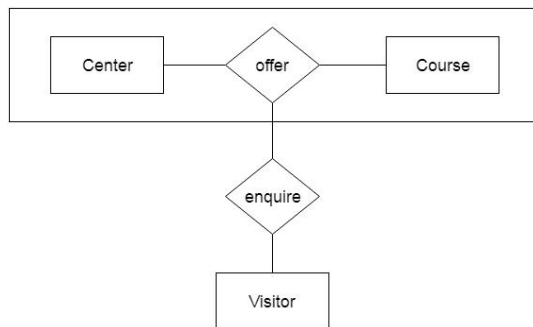
For example: In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.
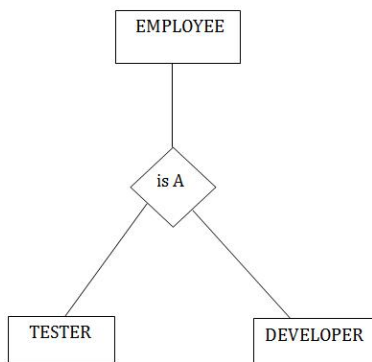
Aggregation

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

For example: Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.