Basic Indexing Techniques in Relational Database

Vinitha C[1], Soumya Unnikrishnan[2], Jinesh V N[3]
[1,2] *Dept. of CA, Tjohn College, Bengaluru*
[3]*Dept. of CS&A, BU, Bengaluru*

***Abstract*- Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing is used to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. This work focuses on various indexing methods used in RDBM files to enhance and optimize the query processing.**

***Index terms*- Indexing, Dense indexing, Sparse Indexing**

I.INTRODUCTION

Any operation in database causes operations on the records where the database is stored. Millions of data transactions taking place every second, database optimization is a key area of optimization and hence the area of research also. Lack of database optimization in relational databases may result into significant costs to both the client and the developer. This paper presents the various database indexing techniques used in commercial DBMS for the optimization of the databases operations.

II. INDEXING OF RECORDS

Database is stored as collection of files. Files can be of two types

A. Primary File

Primary File contains the actual data. Database is a collection of relations. One relation may be stored in a single file or may be stored in several files. One file can also store multiple relations.

B. Secondary File

Secondary File also called auxiliary file or indexed file which is provide an data structure to access the data efficiently. The primary file is stored in secondary storage and size is too large .Indexing is used to improve the performance of a database by reducing the number of disk access. The index created on some data base column it may be key or non-key. The structure of index is as shown below

Search Key	Data Reference /Address

Fig.1 Structure of index [1]

The search key contains the copy of primary key or candidate key or may me any non-key attributes.

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.

Index can be single level or multi-level.

In single level indexing index file maps directory to the block or the address of the record.

Multi-level index has multiple levels of indirections among indexes.

The Index file consists of records with two fields search key and data reference. This file smaller than the primary file. Index file can be dense or sparse. In dense indexing for each record in the primary file one entry is created in the index file. The number of entry in the index file is same as the number of records in the primary file.

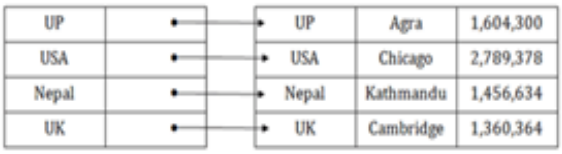

Fig. 2. Dense Index [1]

In the above figure the first table is index file and the second table is primary file. The index is created based on the first column in the primary file. The index file contains the copy of the first column and created a link to each record in the primary file. This kind of indexing is called dense indexing.

In sparse indexing the index file will not have the entry for all the records in the primary file. It will create the mapping for only few records. So the

number of entries in the index file is less than the number of records in the primary file.

Fig. 3 Sparse Index[1]

In the above figure the index file contains only few entries and it is not created the link to all records of the primary table. These kinds of indexing are calls sparse index.

Indexing Methods

1. Ordered Indices
2. Primary Index
3. Secondary Index
4. Clustering Index

1. Ordered Indices

The indices are normally sorted to make the searching faster such kind of indices is called ordered indices.

2. Primary Index – The records in the blocks (small unit in the secondary storage) are stored in the sorted order of the primary key. Index is creating on this primary key i.e the search key is primary key and which is ordered. Since we are using the primary key which is unique and sorted so the index file contain the key of the first record in each block. Remaining records can be accessed based on the first record. So this index is sparse and the number of entries in the index file is equal to the number of blocks used to store the records.

3. Clustering Index – The secondary index is created on a ordered non key attribute .Since it is non-key the values may be duplicated it is not unique. So the records are stored in the sorted order of the non key attribute. The index file creates an entry for each unique search key. So the number of entries in the clustered index file is equal to the number of distinct non key attribute. The non key may be repeated and we are not creating the entry in index file .Hence it is sparse indexing.

In the below figure Dept Id is a non-key and it contains duplicates. The first block contain dept id 1 and 2. The index file contains only distinct dept id and created the link to the block which started storing dept id. Here all the records are sorted.

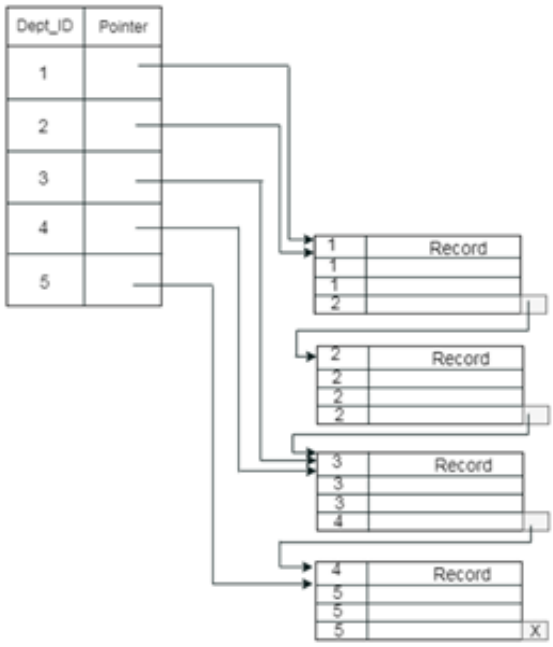

Fig. 4 Clustering Index[1]

Another Implementation of cluster indexing is shown below

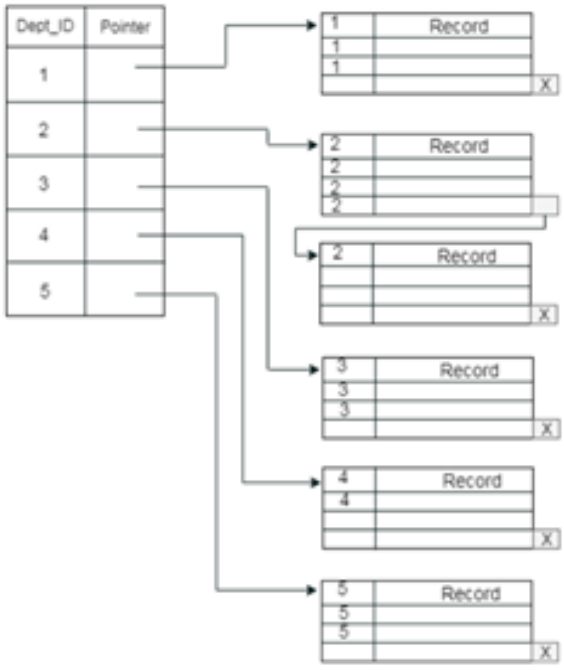

Fig. 5 Clustering index [1]

Here index file contains only distinct values and link is created for the first block which contains the value. Here one each block contain only one value. The second block is full then the same dept id is stored in

the third block and created a link from second block to third. The first and second implementation is dense.

4. Secondary indexing – can be created on either using key or using non key attribute but which is not in order. The record is not stroed in any order. The secondary indexing can be dense or sparse.

Secondary index using key attribute –In this case since we are using key and the records are not in order , the index file create the entry for each key value and corresponding address. Hence it is a dense indexing, and the number of entries in index file is equal to the number of records in the primary file.

Secondary index using non key attribute –In this case since we are using non key attribute and the records are not in order , the index file create the entry for each key value and corresponding address. Then the index file will be sorted. Hence it is a dense indexing, and the number of entries in index file is equal to the number of records in the primary file. Here the index file contains many duplicate and even the index file is sorted the binary search algorithm also will not give better performance. The below figure shows another method

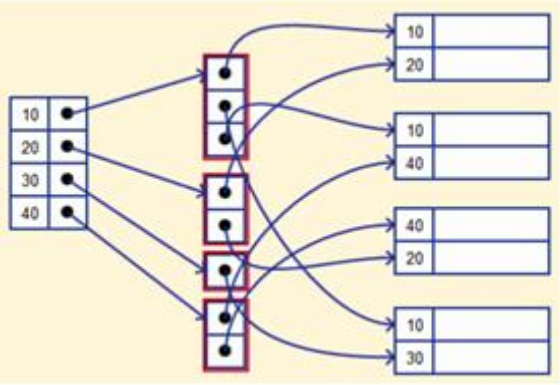

Fig. 6 Secondary index [2]

Here we have primary level index and secondary level index. The primary level index file will contain the entry for each distinct search key and the pointer (address) is pointing to Secondary level which contains the actual address of all duplicates which points to the actual physical location. The number of entries in the index file is equal to the number of distinct search key and which is sorted. Hence this indexing is sparse.

III. SUMMARY AND CONCLUSION

Indexing is a collection of data entries plus a way to quickly find entries with given search key values. Indexing should be used in databases where selection queries are frequent. Indexing should be done on large databases where retrieval of data is performed very frequently. After gone through the various indexing techniques it is found that:

- Hash indexing provides best response time while working on individual keys.
- Bitmap indexing is best suited with low cardinality of key values.
- B-tree indexing works well in range and equality comparisons.

But, the B-tree indexing is widely used in the database systems as majorly queries are based on the combination of range and equality comparison.

IV. CONCLUTION

Indexing is method of data entries to quickly find data with given search key values. If selection queries are frequent then Indexing should be used. Indexing must be done on large databases where retrieval of data is performed very frequently.

A detailed study should be done about indexing and advanced data storage techniques such as block chaining. Efficiency of these indexing in such techniques.

REFERENCE

[1] https://www.javatpoint.com/indexing-in-dbms
[2] https://www.guru99.com/indexing-in-database.html