International Journal of Engineering Research and Development
e-ISSN : 2278-067X, p-ISSN : 2278-800X, www.ijerd.com
Volume 2, Issue 8 (August 2012), PP. 46-50

Optimizing Search for Fast Query Retrieval in Object Oriented Databases Using Signature Declustering

Ms. Ankita Thakur[1], Ms. Meena Chauhan[2]
[1]*Department of Computer Science, JSS Academy of Technical Education, Noida, India*
[2]*Department of Computer Science, ABES Engineering College, Ghaziabad, India*

***Abstract*—Indexing is always been an important question in the efficient information retrieval from the databases. One such approach is the Signature Declustering tree (SD Tree) in Object Oriented Databases (OODB), which has been proved an efficient approach for the insertion and search operations.SD tree is used to represent 1s (set bits) of the signature in the clustered form. This makes the faster retrieval of the information. In our approach we have used the SD tree for storing 0s (unset bits) as well as 1s (set bits) in the clustered form. This makes the search faster for the signatures whose signature weight above 50%.Searching such signatures using the unset bits in the SD tree has improved the search time by 43% approximately. Analytical experiments have been conducted using various signatures and it has been proved that the superiority of the approach.**

***Keywords*— Signature file, Objected oriented query handling, Information retrieval, Signature Declustering**

I. INTRODUCTION

Indexing is a very important question while managing the large amount of the data in the databases. Indexing plays a vital role in the faster recovery of the desired data from the large databases. There are many techniques that have been frequently used for the managing the data in the databases like full text searching, B-trees [2] , inversion [3],[4] and the signature file[5],[6],[7]. Full text searching technique has less space overhead but requires large response time. In contrast, B-tree, inversion and the signature file techniques works quite better but these techniques need a large intermediary representation structure (index), which provides the direct links to the data.[8] Among all these techniques the signature file approach has been proved very efficient for the easy handling of insert and update operations. Signatures are the hash coded abstraction of the original data. It is bit string of pre-defined length with fixed number of 1s.An object's signature can be formed by the superimposition of its attributes signature. The query processing with the signature file is performed in the following manner. : When a query is passed ,its query signature is generated by applying some hash function to its attribute values. Then to resolve the query, this query signature Sq is compared with signature files for the 1s sequentially. If the query signature Sq matches with any of the signature file then that signature file becomes a candidate that mat satisfy the query. If all the 1s in the query signature matches with the 1s in the signature file, it is called a drop. The next step for the query processing is the resolution of the false drops. Each drop needs to pass through a filter test to recognize whether it is a actual drop or a false drop. Drops that fails the test, are called the false drops [5],[6], while the qualified ones are called the actual drops.

II. RELATED WORK

This section discusses about the various approaches used for signature file representation and a brief summary of the signature file approach.

A. BACKGROUND

There are various approaches that have been used for the signature file representation such as Sequential Signature file, Bit Slice signature file,Multi-level Signature file, Compressed multiframed signature file, parallel signature files-tree and its variants signature graph, signature tree and the SD-Tree [1].

In this paper we have the studied the advantages that could arise from storing the 0s in the SD-tree. In the SD –tree ,all the set and unset bits are distributed over various signature nodes and all the signatures having a common bit are clustered together, so that a query processing is fast and brings all output signatures in a single access.

In the SD-tree , we are storing both the set and unset bits and distributed them across the various signature nodes. Then all the signatures having a common bit are clustered together so that a query can be processed in a single access with the minimum processing time .

B. SUMMARY OF SIGNATURE FILE APPROACH

A signature of an attribute is an hash coded bit string with certain set bits. There are various approaches for the index structure but the signature file has proved to be the more efficient in handling the search and the insertion operations. The other advantages of the signature files are that, they are easier to implement and their ability to support the growing file. Signature extraction method should be chosen carefully to minimize the information loss.

An object signature can be generated by superimposing its all attribute signatures. Superimposing means simply performing the bitwise OR operation. Object signatures for a class is stored sequentially in a file, called a Signature File.

When a query for a signature arrives the object signatures will be searched sequentially for the match. All the non qualifying signatures will be discarded. Query signature Sq is the nothing but the signature generated for the certain values as the way generated for the attribute values. Then each object signature is searched for the match sequentially. An example of the sample query evaluation is explained below:-

EXAMPLE:-

Information	**0010 0100**
Retrieval	**0100 0001**

Sample Queries

- **Matching Query**

Keyword = Information	**0010 0100**
Query Descriptor	**0010 0100**

Block Signature Matches (Actual Drop)

- **False Match Query**

Keyword = Coding	**0010 0001**
Query Descriptor	**0010 0001**

Block Signature Matches (False Drop)

- **Non Matching Query**

Keyword = Information	**0010 0100**
Keyword = Science	**0000 0110**
Query Descriptor	**0010 0110**

Block Signature does not match

III. SD TREE STRUCTURE

SD – tree is the signature declustering tree, using the dynamic balancing over B+ tree. The SD-Tree stores all the set bits in the clustered manner to reduce the insertion and the search time for the queries. The structure of the SD-tree can be explained as:-

It has three kinds of the nodes.

- Internal Node
- Leaf Node
- Signature Nodes

The Internal nodes and the leaf nodes are quite similar to the structure of the internal and the leaf nodes of the B+ tree respectively. The upper level of the tress is comprises of the internal nodes. The leaf nodes are at the one level above the last level. The signature nodes are at the last level of the SD-tree. To illustrate the structure of the SD-Tree we have assumed the order of the tree as three and the signature length as 12 bits.

A. STRUCTURE OF THE INTERNAL NODE

An internal node of the SD –tree is similar to the B+ tree, consists of the pointers and the keys. For the tree of order three, the internal node has two keys K1,K2 and the three pointers P1,P2, and P3.Each pointer is pointing to the nodes at the lower level. The left pointer is pointing to the values less than or equal to the node values while the right pointer is pointing to the values greater than the node values as in the B+ tree.

P1	**K1**	**P2**	**K2**	**P3**

Figure 1. Structure of the Internal Node.

B. STRUCTURE OF THE LEAF NODE

The leaf node appears at the one level above the last level. The key values appears in the ascending order of their values. But unlike B+ tree SD-tree has no sequential pointers between the leaf nodes instead each leaf node is pointing to a chain of signatures. Pointer P1 and P2 is pointing to corresponding signature nodes for key value K1 And K2.

K1	**P1**	**K2**	**P2**

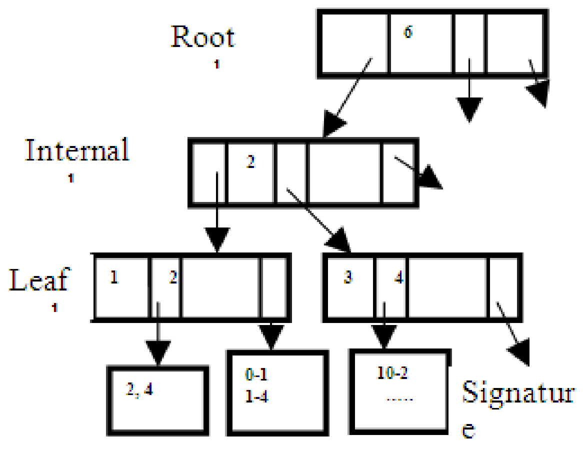
Figure 2. Structure of the Leaf Node.

C. STRUCTURE OF THE SIGNATURE NODE

The signature node for Ki has 2 i-1 binary combinations denoting the possible prefixes. When the signature Su with the 1 in the ith position is to be inserted then its intermediate generated prefix is to be compared with the binary combination in the signature node with key value Ki and u is inserted in the list.

D. THE COMPLETE SD TREE

The tree of order three has been constructed for the signatures of length 12 bits The root node has the key value K1 as 6.At the next level the internal node has the value 2 as key value of K1 which is pointing to the leaf nodes with value 1 and 2.Consider the insertion of the signatures from the signature file, the first signature S1 is read after the tree construction for the given signature length is over. The S1 has its first set bit at position 2 hence the leaf node with the key value 2 is accessed and its signature node is followed. The possible prefixes for the signature node 2 are all 0 and 1. Hence S1 is inserted in the prefix list and if the list does not exist a prefix is generated and the signature value 0-1 is inserted. For the signature S2 having set bits at position 1 and 3 the signature value 2 is inserted at signature node 1 with no prefix and with prefix 10 at signature node 3. [1]



(a)

S1: 010…….
S2: 101…….
S3: 000…….
S4: 110…….

Figure 3. Structure of the SD tree.

The storing of binary prefixes makes the query processing faster. Consider the search for the query signature Sq using the SD tree. For example consider Sq=011001000101 . To find all the matching signatures for Sq the tree is traversed from root and the node values are compared with bit positions of Sq.The last occurrence of 1 in Sq is at position 12. The binary prefix generated for Sq using the position of 1s is 01100100010. A node with the key value 12 is accessed in the signature list of storing 1s in clustered form and all the signatures in the signature list is checked for the prefix value 01100100010. Hence regardless of the bit pattern of Sq all the matching signature are returned in a single access.

Similarly , we have used SD tree is to represent 0s in the clustered way. This will improve the search time for the signatures having signature weight greater than 50%. Now, assume Sq = 011101110101.The last occurrence of 0 is at 11th position. So the node with the node value 11 is searched in the signature list storing 0s in the clustered form.

IV. SD TREE INSERTION AND SEARCH

This section gives the algorithm for the signature insert and search operations.

A. INSERTION

Insert (Su)
Input: The signature to insert Su; u is the signature number

```
    1.  Move the set bit positions of Su in an arraylist1 and for the unset bit in arraylist0.
    2.  For each value i in queue do
Begin
Access leaf node i;
Write the signature no. with the prefix;
End.
```

B. SEARCH

Search (Sq)
Input: The (query) signature to search.

Output: The list of signatures matching the given signature.

1. Compute the Signature weight for the query signature.
2. If signature weight is greater than 50% then search the query signature in the leaf nodes for the unset bits.
3. Else search the query signature in the leaf nodes for the set bits
4. Access leaf node.
5. Compare the prefix of Sq.
6. If Found () then read and output the list of signatures.
7. Else report "no matching signatures".

V. RESULTS AND DISCUSSION

To analyze the results of the proposed approach we implement the SD tree using Java programming language and run simulation experiments on various signatures. The experiments were performed on a stand alone system with Pentium core i3 processor, 2.40 GHz with 3GB RAM, and 320GB hard disk.

A. TIME COMPLEXITY

Considering the response time as a performance measure, we have compared the search complexity for the signature with signature weight above 50% using the SD tree storing 1s and 0s in clustered form. Performing search operations using the unset bits for the signatures with above 50% signature weight has proved to be efficient as it takes lesser amount of time as compared to set bits approach. The parameters used are Signature length, signature weight and time in nano seconds. The graphical representation of the insertion time taken by the signature for inserting in the SD Tree has been illustrated by the fig given below.

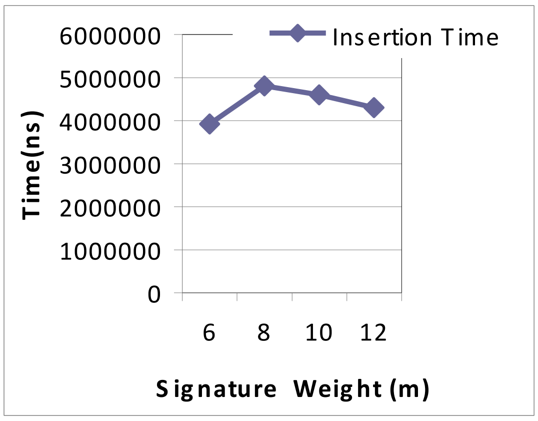


Figure 4. Insertion Time for the Signatures.

The above given fig 4. shows that signature with the signature weight 6 is taking 3921354 nano seconds to get inserted in the SD Tree at desired location. Searching the query signature with signature weight above 50% using the unset bits has improved. the query search time by 43% approximately. The graphical comparison of the query search time using both set and unset bits for the various signature weights has been illustrated by the fig 5.

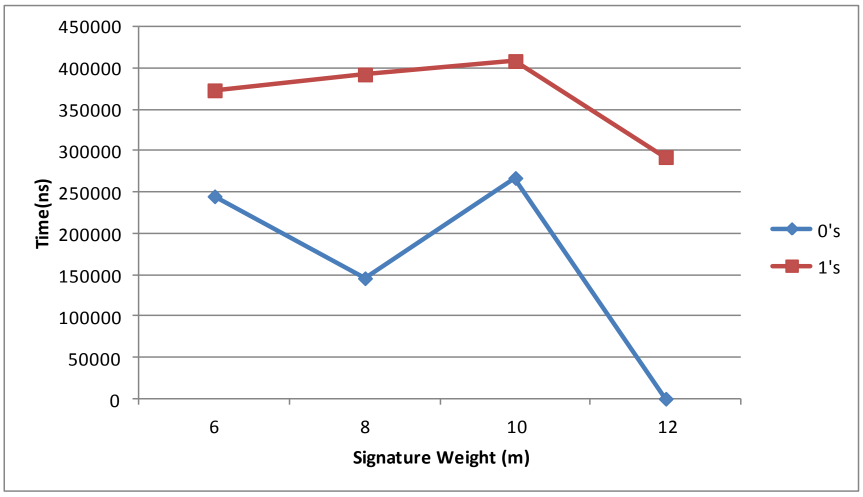


Figure 5. Query Search time for the Signature using set and unset bits.

VI. CONCLUSION

In this approach the SD tree is used to represent 1 and 0s of the signatures in the clustered form. The performance of both the approaches has been analyzed for the searching operations of the signatures with the signature weight above 50%.The analytical results has proved the superiority of the search approach using unset bits as search time has been improved .

The space overhead is higher for the SD tree because of the storage of the prefixes in the higher order signature nodes. Although for the proposed methodology the space complexity becomes quite high as we have to store both the set and unset bits in the clustered form in the SD tree, but storing the unset bits makes the search complexity quite better for the signatures with higher signature weights.

REFERENCES

[1]. A.J. Kent, R. Sacks-Davis, and K. Ramamohanarao, "A Signature File Scheme Based on Multiple Organizations for Indexing Very Large Text Databases," *J. Am. Soc. Information Science*, vol. 41, no. 7, pp. 508-534, 1990.

[2]. Bertino E, Kim W, " Indexing techniques for queries on nested objects *", IEEE TKDE*, Vol.1, No.2, pp .96-214, 1989

[3]. C. Faloutsos, "Access Methods for Text," *ACM Computing Surveys*,vol. 17, no. 1, pp. 49-74, 1985.

[4]. C. Faloutsos, "Signature Files," *Information Retrieval: Data Structures & Algorithms*, W.B. Frakes and R. Baeza-Yates, eds., pp. 44-65, New Jersey: Prentice Hall, 1992.

[5]. Chen,Y. and C. Yibin,2006. " On the signature tree construction and analysis*." IEEE Trans*. Knowledge Data Eng.,18:1207-1224.

[6]. Chen,Y.,2002," Signature Files and Signature Tress." *Inform. Process. Lett*.,82:213-221

[7]. Christos Faloutsos, " Access Methods for Text ", *ACM Computing surveys*, Vol. 17, No. 1, Mar 1985, 49 – 74.

[8]. D. Harman, E. Fox, and R. Baeza-Yates, *"Inverted Files,"Information Retrieval: Data Structures & Algorithms,* W.B. Frakesand R. Baeza-Yates, eds., pp. 28-43, New Jersey: Prentice Hall,1992

[9]. Dik Lun Lee, Young Man Kim, Gaurav Patel, " Efficient signature file methods for text retrieva l" , *IEEE TKDE* , Jun 1995, Vol.7, No.3, 423-435.

[10]. Eleni Tousidou, Alex Nanopoulos, Yannis Manolopoulos, "Improved Methods for Signature-Tree Construction", *The Computer Journal*, 2000, Vol. 43, No. 4, 301 – 314.

[11]. Eleni Tousidou, Panayiotis Bozanis, Yannis Manolopoulos, "Signature-based structures for objects with set-valued attributes", *Information Systems*, Vol. 27, No. 2, 2002, 93 – 121.

[12]. I. Elizabeth Shanthi, Y. Izzaz, R. Nadarajan, "On the SD-tree construction for optimal signature operations", *Proc. of ACM COMPUTE , ACM Digital Library* ,Jan 2008.

[13]. J. Zobel, A. Moffat, and K. Ramamohanarao, "Inverted Files Versus Signature Files for Text Indexing," *ACM Trans. Database Systems*, vol. 23, no. 4, pp. 453-490, Dec. 1998.

[14]. Kemper A, Moerkotte G ,"Access support relations: an indexing method for object bases", *Inform Sys* Vol. 17, No.2, pp. 117–146. 1992,

[15]. Kjetil Norvag "Efficient Use of Signatures in Object-Oriented Database Systems", *Proc.of Advances in Database and Information Systems ADBIS'99* ,Sep, 1999.

[16]. R. Bayer and K. Unterrauer, "Prefix B-Tree," *ACM Trans. DatabaseSystems*, vol. 2, no. 1, pp. 11-26, 1977.