CHAPTER 4

Chapter 4

Inclusive query processing

The use of signature files as an index for full text search has been widely known and used. Signature file based access methods initially applied on text have now been used to handle set-oriented queries in OODBSs. Today, most of the stored data consists of records with set-valued attributes. Since OODBSs handle efficiently multi-valued attributes they are widely used. The basic query for this kind of data is the inclusive or partial match query that retrieves all records containing specific attributes.

4.1 Signatures and inclusive queries

Authors of [HELMER 03] report that signatures are preferred to encode sets for three reasons.

- They are of fixed length and hence convenient for index structures.
- Set comparison operators on signatures can be easily implemented by efficient bit operations.
- Signatures are more space efficient compared to the conventional set representation.

Inclusive queries are further divided into subset and superset queries [TOUSIDOU 02]. Subset queries retrieve objects with set-valued attributes that contain the query set. In superset query, objects whose set-valued attribute are contained in the query set are retrieved. In other words subset queries match all signatures with matching 1s *at least* in the query signature whereas superset queries match signatures with *at most* 1s (or its subsets) of the query signature. We follow the lead of [TOUSIDOU 02] for background information on inclusive queries. Consider the object schema with classes having set-valued attributes (partially taken from the sample object schema of Figure 3.10) shown in Figure 4.1.

The classes with set-valued attributes are Dept, Instructor and Programme. Every object of Dept has a list of Pgm-name offered in its attribute Programme. Similarly every Programme object has a list of Course-name and each Instructor teaches one or more Course-name denoted in the attribute Course.

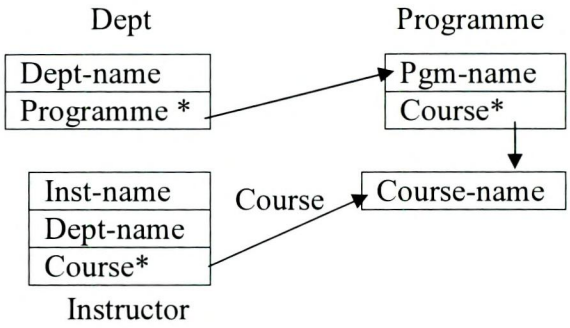

Figure 4.1 Schema of classes with set-valued attributes

For any query signature O' and an object O satisfying the query, subset query searching is denoted as

$O' \subseteq O \Rightarrow q \subseteq s$ and for superset query $O' \supseteq O \Rightarrow q \supseteq s$.

The right hand side of both equations denotes that the corresponding query (q) and object signature (s) also satisfy the same.

4.2 Query Algorithms

The SD-tree which retrieves signatures efficiently for exact query matching can be easily adapted to handle subset-superset queries too. This section outlines the algorithms for processing subset and superset queries using SD-tree.

4.2.1 Algorithm Subset

Subset query processing search for all signatures whose set bit positions match with that of query signature. In SD-tree based retrieval the output is the common elements of all signature nodes at the set bit positions of query signature. The steps of the algorithm are shown in Figure 4.2.

Algorithm Subset

1. Let S_q be the given query to search for.
2. Let i_1 , i_2, … i_n be the positions of set bits in S_q.
3. Move i_1 , i_2, … i_n into a queue. Let j = 1.
4. While queue not empty do

Begin

Read i_x from queue.

Access leaf node i_x ->sig.node;

Let S_j = Set of all sig. numbers in the sig. node;

j = j+1;

End.

5. Result = $S_1 \cap S_2 \ldots \cap S_n$.
6. {Result} has all signatures matching S_q.

Figure 4.2 Algorithm Subset

Generally, in the subset query handling process, for a given query signature S_q, all signatures from signature file having set bits at least in the set bit positions of S_q are retrieved. The same process can be easily implemented in the SD-tree structure too. For each set bit position of S_q, the algorithm retrieves cumulatively all signatures from the corresponding signature node and moves them to separate lists. The signatures that occur in all the lists (intersection of all sets) are the qualified ones. This is because such signatures either match exactly with S_q or having set bits at least in the corresponding positions of S_q. This way matching subset queries using SD-tree is simplified.

4.2.2 Algorithm Superset

Superset query contains the maximum number of set bits to search. Hence, the algorithm finds all signatures whose set bits form subsets of the query. The steps in the algorithm are listed in Figure 4.3.

In superset queries the matching signatures are the ones having at most 1's as in the query signature. In other words signatures having set bits in positions which are either the same as that of S_q or a subset of 1's in S_q are the qualified signatures for output. This can easily be implemented using SD-tree. The algorithm initially forms the signatures of the possible subset values from the attributes that are part of the query. Then these signatures are used to find the matching signatures (by comparing the binary prefix too) and moved to output lists. These lists ensure signatures having partial match with S_q. Now the signatures present in all the sets are the required output.

```
Algorithm Superset
        1. Let Sq be the query signature.
        2.  Let n be the number of attributes superimposed to
              form Sq .
         3. Move all possible subsets (2n -1) (superimposed
              signatures of attribute combinations except null
              set) into the queue.  Let i = 1;
         4. While queue not empty do
              Begin
              Read signature x from queue;
              Access sig. node of last set bit of x.
              Let B be the binary prefix of the current
              sig. node considered one at a time.
              For all prefixes in sig. node do
              Begin
               If Sq.prefix ∩ B = =  Sq.prefix then
               Si = Set of sig. no.s pointed by B.
               i = i+1;
               End.
         5. Result = S1 ∩ S2 ….∩ Sn.
         6. {Result} has all signatures matching Sq.
```

Figure 4.3 Algorithm Superset

From Figure 4.2 and Figure 4.3 we can infer that the SD-tree which is retrieving matching signatures for a given query signature in a single access is performing in a consistent manner for subset and superset queries also.

4.3 Experimental results

To run queries we implement SD-tree in Java and for every test run the tree is constructed statically before signature insertion. The parameters considered in the experiments' data sets are signature length (F), and the number of set-value attributes considered in the query (s). The experiments were carried out in the same hardware set up mentioned in Section 3.4. We considered for experiments abstract data set having 60000 objects fixed for various signature lengths.

4.3.1 Time Complexity

Like in other signature applications we use the response time as the performance measure. The time complexity of handling subset queries depend on the number of set bits (m) in the query signature. The search time within the signature node of set bits depend on the average number of entries (a). Hence, the complexity of subset queries is bounded by O(ma). The number of values in the set is varied from 2 to 4 for signature lengths 8, 12 and 16 and the noted readings are listed in Table 4.1 and values are plotted in Figure 4.4.

In order to improve selectivity of signatures for varying number of attributes in the query, signature length was varied to keep false drop probability at minimum. It is obvious that the number of values of the set-attribute considered in the query definitely increase the set bits in the output signature increasing the searching time. Otherwise the time variations are only approximate as signatures are mere abstraction of original

attribute values and superimposing attribute values further vary the number of set bits in the output object signature.

Table 4.1 Subset query evaluation time (in Sec)

Subset query			
Signature length (F)	Set value		
	2	3	4
8	0.002	0.01	0.01
12	0.01	0.02	0.02
16	0.01	0.05	0.07

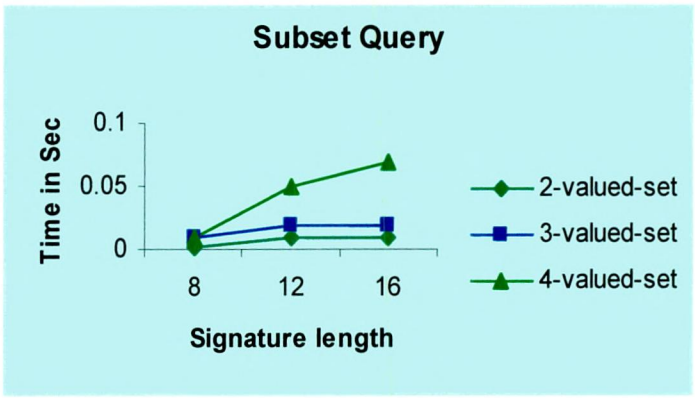

Figure 4.4. Subset Query

For superset queries the algorithm considers the subsets of the values of the set-attribute (except null set) that vary from 2 to (s). Obviously the time taken for query searching increases with (s). This is shown in Fig 4.5 and the observed time values are listed in Table 4.2. Searching within the signature node proceeds as in SD-tree but for $(2^s - 1)$ subset values. It is apparent that the time taken for query searching increases with the constituent signatures of (s). If w is the number of subsets of s, then the time complexity is bounded by O(wa). This value is substantially less compared to the signature tree search cost that is dependent on the signature file size N.

Table 4.2 Superset query evaluation time (in Sec)

Superset query			
Signature length (F)	Set value		
	2	3	4
8	0.01	0.03	0.04
12	0.02	0.04	0.05
16	0.03	0.04	0.05

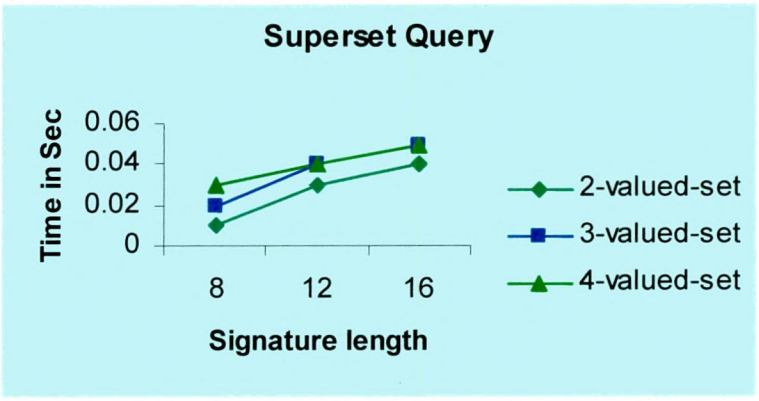

Figure 4.5 Superset Query

4.3.2 Space overhead of SD-tree

The space overhead is the same as that of the SD-tree discussed in section 3.4.2. This due to the integration of SD-tree adaptations for handling subset-superset queries in the algorithmic steps only. So, the tree structure is unaltered and hence the space complexity.

4.4 A sample validation model

A list of sample subset-superset queries with the class accessed and the matching signatures that are output is shown in Table 4.3. For the schema shown in Figure 4.1 we make example queries bound to classes Programme and Instructor which have set-valued

attribute "Courses". The signature files of classes are assumed to be same as that of Table 3.4.

Table 4.3 Sample inclusive queries

Criterion	Class accessed	Query signature (Sq)	Result
Query 1 : List of programmes offering atleast Comp-applns in Courses			
Course-name = Comp.applns	Programme	00101000	{1,3}∩{1,2,3}= {1,3}= M.E, M.Sc (Phy)
Query 2 : Instructors handling atleast Software engg in Courses			
Course-name = Software engg	Instructor	01000001	{1,3,4}∩{1,2,3}= {1,3}=John, James
Query 3 : Programmes offering Courses none other than Comp. applns and Applied Maths			
Subsets are Comp. applns(CA) AppliedMaths(AM) CA & AM	Programme	00101000 10000001 10101001	{1,3}∩{1,2}∩{1} = {1} M.E
Query 4 : All instructors handling none other than Applied Maths and Calculus			
Subsets are AppliedMaths(AM) Calculus (CAL) AM & CAL	Instructor	10000001 00100100 10100101	{1,2}∩{2}∩{2}= {2} Adams

In Table 4.3, Query 1 and Query 2 are subset queries whereas Query 3 and Query 4 are of type superset. For example, Query 1 requires all programme names for which

Comp. applns is a subset of Courses attribute. As per algorithm in Figure 4.2, the two set bits of the query signature which are at positions 3 and 5 are moved to a queue and all matching signatures with set bits at the same position is extracted as S1 and S2. The intersection of these sets is the required result (i.e) M.E and M.Sc (Phy).

Similarly superset query (for example Query 3) requires all programmes having the listed attributes (i.e) Comp. applns and Applied Maths as the superset of the output. In other words the search procedure has to find none other than all matching signatures of the subset of the given set-valued attribute in the query. In Table 4.3, the result of query 3 is the intersection of all resultant sets which is the course M.E. Query 2 and query 4 can be interpreted in a similar way for Instructor class with different combinations for the set-valued attribute Courses.

In this chapter the report on adapting SD-tree to support subset-superset queries and analyzing the performance for query response time has been presented. The proposed system is a simple and flexible indexing structure to represent signature files. It supports efficient handling of object-oriented inclusive queries on set-valued attributes.