EAI Endorsed Transactions
on Context-aware Systems and Applications Research Article ICST.ORG

Querying Object-Oriented Databases Based on Signature Graph and n-ary Tree

Tran Minh Bao[1,*], Truong Cong Tuan[1], Huynh Trong Duc[2]
[1] College of Sciences, Hue University, Vietnam.
[2] Ho Chi Minh City Industry and Trade College, Vietnam.

Abstract

In this paper, we suggest a new technique to create index helping for querying almost identical similarities with keywords in case there is no correct match found. It's based on a signature graph and n-ary tree helping to query related information when there is no correct match. Main idea is a signature graph structure, created based on signature file, created for a layer and signature files are arranged as a hierarchical system according to nested structure of layers. This technique helps to decrease effectively search space, so therefore it will improve effectively complexity of query time. More than that, we develop query algorithm on signature graph based on Chen and partners method [3] suggested, thereby helping to improve query time on signature graph better.

Keywords: Object-oriented query, object signature, signature file, signature graph.

Received on 21 December 2015, accepted on 11 January 2016, published on 12 February 2016

doi: 10.4108/eai.12-2-2016.151087

1. Introduction

Study of indexing technique is always an important issue in effective information searching from databases. For object-oriented databases, direct query on objects has a large time cost. There are many database indexing techniques to process query on object-oriented databases in which signature file approach has been widely acknowledged and been an effective approach in processing query on object-oriented databases. For this approach, objects of a class are coded into object signatures by using hash function and stored in a signature file. However, query on signature file has a disadvantage which is high cost due to scanning the whole file. Some other indexing methods try to overcome this and can be found in many researches [2, 3, 4, 9, 10].

Various indexing techniques for object oriented databases specifically for aggregation hierarchy are summarized in [1] and it is shown that indexing technique based on signature file has the best performance for the retrieval operation.

Signature tree [2, 3] and signature graph [3, 4] are improved indexing techniques based on signature files. Signature tree which is based on signature file arranges all signatures in a tree form, which leads to less comparison and improves query evaluation. Signature graph is an indexing technique built for a class and in addition arranges signature files in aggregation hierarchy and improves the search of a signature file.

In this paper, we propose improvement of query algorithm on signature graph which can be used to improve query time. Firstly, we organize sequential signature files in nested signature file hierarchy to reduce searching space during querying. Then we store each signature file in signature graph or n-ary tree to speed up signature file scanning. The larger signature file is, the more time can be saved by using this approach.

This paper is organized as follows. In part 2, we provide background. Part 3 proposes indexing technique. Part 4 proposes an approach combining signature file hierarchy and signature graph. Finally, part 5 gives out a conclusion.

*Corresponding author. Email:tmbaovn@gmail.com

2. Background

2.1. Attribute Signature

In an object-oriented database, each object is presented by a set of attribute values. Signature of an attribute value is a sequence of hashed-code bits. Given an attribute value, for example the word "student", we decompose it into a string of three-letter sets as follow: "stu", "tud", "ude", "den" and "ent". Then, using hash function h, we map a triplet to an integer k which means kth bit in a string assigned value 1. For example, assuming that we have h(stu) = 2, h(tud) = 7, h(ude) = 10, h(den) = 5 and h(ent) = 11. Then we create a bit string: 010 010 100 110 which is signature of the word.

2.2. Attribute Signature, Signature file

Object signature is constructed by logical OR algorithm for all signatures of attribute values of the object. Below is an example of an attribute signature:

Example 1. Consider an object which has attribute values of "student", "12345678", "professor". Suppose that signature of these attributes is:

010 010 100 110
100 010 010 100
110 100 011 000

In this case, object signature is 110 110 111 110, generated from attribute signatures by using logical OR algorithm. Object signatures of a class are stored in a file, called object signature file.

2.3. Query Signature

An object query will be encoded into a query signature together with hash function applied to objects. When a query needs to be executed, object signatures will be scanned and unmatched objects will be excluded. Then query signature is compared with object signatures of signature file. There are three possibilities:

(i) The object matches with the query, i.e., for every bit in query signature s_q, corresponding bit in object signature s is the same, i.e, $s_q \wedge s = s_q$, a real object of query.

(ii) The object does not match with the query, i.e., $s_q \wedge s \neq s_q$;

(iii) Signatures are compared and matching one is found but its object does not match with searching condition of the query. To eliminate this case, objects must be checked after object signatures are matched.

Example 2. This example illustrates the query for object signature in example 1:

Query:	Query signature:	Result:
student	010 000 100 110	successful
john	011 000 100 100	unsuccessful
11223344	110 100 100 000	false drop

Comment: comparing query signature s_q object signature s is incorrect comparison. That means, query signature s_q matches with signature s if for any 1 bit in s_q, the corresponding bit in s is also 1 bit. However, for any 0 bit in s_q, the corresponding bit in s can be 0 or 1.

2.4. Querying Object-Oriented Databases

In object-oriented database systems, an entity is represented as an object, which consists of methods and attributes. Objects having the same set of attributes and methods are grouped into the same class. Since a class C may have a complex attribute with domain C', a relationship can be established between C and C'. The relationship is called the aggregation relationship. When arrows connecting classes are used to represent the aggregation relationship, an aggregation hierarchy can be constructed to show the nested structure of the classes.

Example 3. An example of a nested object hierarchy:

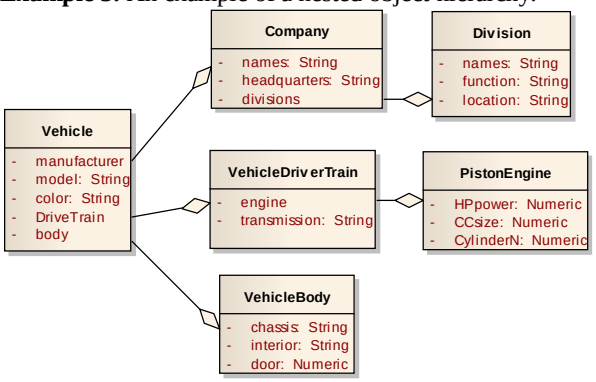

Figure 1. An example of a nested object hierarchy

If an object o is referenced as an attribute of object o', then o is said to be nested in o', and o' is referred as the parent object of o.

In object-oriented databases, the search condition in a query is expressed as a combination of attribute. The attribute may be a nested attribute of the target class.

Example 4. The query "retrieve all red vehicles manufactured by a company with a division located in Ann Arbor" can be expressed as:

select vehicle
where Vehicle.color = "red"
and Vehicle.company.Division.location = "Ann Arbor"

Without indexing structures, the above query can be evaluated in a top-down manner as follows. First, the system has to retrieve all of the objects in the class Vehicle and single out those that are red in color. Then, the system retrieves the company objects referenced by the red vehicles and checks the locations of the divisions of the manufacturers. Finally, those red vehicles made by a company that has a division located in "Ann Arbor" are returned.

2.5. Signature File Hierarchy and Query Algorithm

• *Signature File Hierarchy*

The purpose of using a signature file is to screen out most of the nonqualifying objects. A signature failing to match the query signature guarantees that the corresponding object can be ignored. Therefore, unnecessary object accesses are prevented. In terms of an aggregation hierarchy, a signature file hierarchy can be constructed as follows:

(i) The signature of an object is generated by superimposing the signatures of all its primitive and complex attributes.

(ii) The signature of a primitive attribute is obtained by hashing on the attribute values; the signature of a complex attribute is the signature of the object it references.

(iii) Let C be a class, and let o_1, ...,o_l be its objects; there exists a signature file S such that each o_i(i= 1, ..., l) has an entry <osig, oid> in S.

(iv) Let S_i and S_j be two signature files associated with classes C_i and C_j, respectively. If there exists an arrow from C_i to C_j, then there is implicitly an arrow from S_i to S_j.

Example 5. Signature and signature file hierarchy:

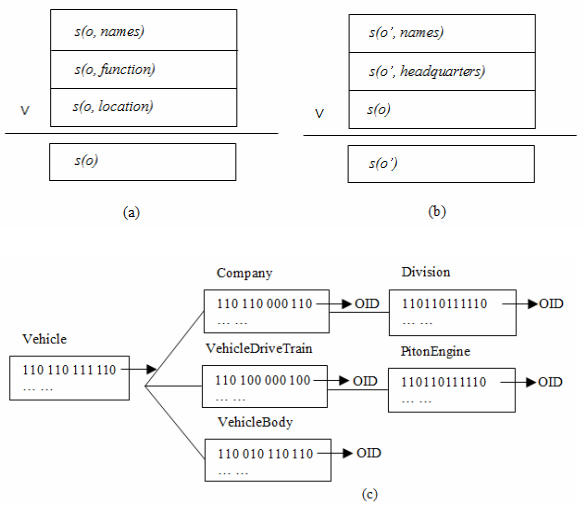

Figure 2. Signature and signature file hierarchy

Consider the class "Division" in the class hierarchy shown in figure 1, which contains no complex attributes. The signature of an object o of this class can be constructed as shown in figure 2 (a), where each s(o, x) stands for the signature produced for the attribute value x of o and s(o) for the signature of o. For a class containing complex attributes, the signature of its objects can be generated in the same way as for a class containing only primitive attributes. The only difference is that the signature of a complex attribute is the signature of the object it references. See figure 2 (b) for an illustration. In figure 2 (b), o' stands for an object of class "Company", and object o of class "Division" is the attribute value of "division" of o'. Signature file hierarchy may be constructed for a database with the schema shown in fig 1 for an illustration in fig 2(c).

• *Query Algorithm Based on Signature File*

Definition 1. (Query tree) [3] Let $p_1 \wedge p_2 \ldots \wedge p_k$ be the search condition in query Q, where each p_i is a predicate of the form: <attribute operator value>. Then, all the paths appearing in the search condition constitute a query tree, denoted as Q_t.

Example 6. Query tree:

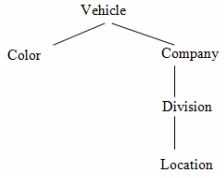

Figure 3. Query tree

Definition 2. (Query signature tree) [3] Let $p_1.p_2 \ldots .p_n$ be a path in a query tree Q_t (from the root to some leaves). Let <$p_i \ldots .p_n$ operator value> be a predicate appearing in the search condition of Q. Then p_n's signature is s_{value}. The signature of a non-leaf node in Q_t can be obtained by superimposing the signatures of its child nodes. The query signature tree is denoted as $Q_{(s,t)}$.

Example 7. The query signature tree:

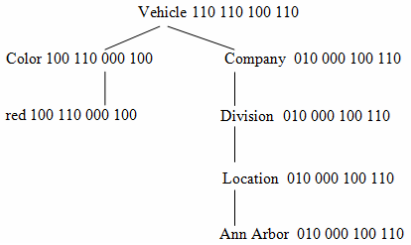

Figure 4. Query signature tree

Use the query signature tree to reduce searching space. For this purpose, two stack structures are needed to control depth-first traversal of tree structures: $stack_q$ for $Q_{(s,t)}$ and $stack_c$ for the class hierarchy. In $stack_q$, each element is a signature, while in $stack_c$, each element is a set of objects belonging to the same class reached during class hierarchy traversal.

Algorithm 1. [3] top-down-hierarchy-retrieval;

Input: an object query Q;

Output: a set of OIDs whose texts satisfy the query.

Method:

Step 1. Compute the query signature hierarchy $Q_{(s,t)}$ for the query Q.

Step 2. Push the root signature of $Q_{(s,t)}$ into $stack_q$; push the set of object OID of the target class into $stack_c$.

Step 3. If $stack_q$ is not empty, $s_q \leftarrow$pop $stack_q$; else go to (7).

Step 4. S$\leftarrow$pop $stack_c$; for each oid_i E S, if its signature $osig_i$ does not compare s_q, remove it from S; put S in S_{result}.

Step 5. Let C be the class to which the objects of S belong; let C_1, ..., C_k be the subclasses of C; then partition the OID set of the objects referenced by the objects of S into S_1, ..., S_k such that S_i belongs to C_i; push S_1, ..., S_k into $stack_c$; push the child nodes of s_q into $stack_q$.

Step 6. Go to (3).

Step 7. For each leaf object, check false drops.

In this technique, optimization is achieved by executing step (4). In this step, some objects are filtered using the corresponding signature in the query signature tree. In step (5), the referenced objects and the signatures of the child nodes of the query signature tree are put in $stack_c$ and $stack_q$, respectively. In step (7), the checking of false drops is performed.

Example 8. Assume that a part of the signature file hierarchy constructed for a database with the schema shown in Figure 1 is of the form shown in the upper part of Figure 5:

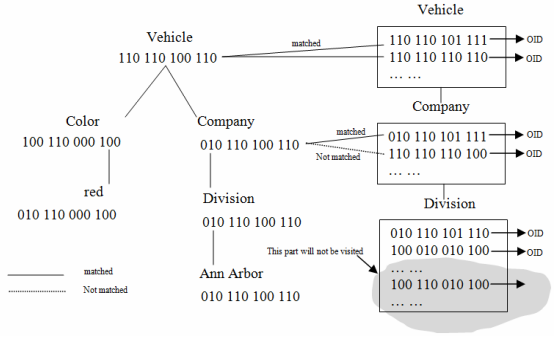

Figure 5. Illustration of query evaluation

Since both the top two signatures in the signature file for Vehicle match the corresponding signature in the query signature tree, the signatures referenced by them in the signature file for Company are further checked. Assume that the first signature in Company is referenced by the first signature in Vehicle while the second one in Company is referenced by the second one in Vehicle. We can see that the second signature in Company does not match the corresponding signature in the query signature tree. Thus, all those Division object signatures referenced by it will not be checked further (see the grey part of Fig 5 for an illustration). This is optimal compared to "top-down-retrieval" since by means of "top-down-retrieval", checking against all Division object signatures has to be performed.

2.6. Signature Graph

• *Construction of Signature Graph*

To find a matching signature, a signature file has to be scanned. If it is large, the amount of time elapsed for searching such a file becomes significant. The first idea to improve this process is to sort the signature file and then employ a binary searching. Unfortunately, this does not work due to the fact that a signature file is only an inexact filter. The following example helps for illustration.

Example 9. Consider a sorted signature file containing only three signatures:

010 000 100 110
010 100 011 000
100 010 010 100

Assume that the query signature s_q is equal to 000 010 010 100. It matches 100 010 010 100. However, if we use a binary search, 100 010 010 100 cannot be found. On the other side, there might be the same signatures in the signature file that match with objects having the same content, query processing needs to find out all locations of suitable objects. Due to this reason, we will organize signature file in a graph, called signature graph, to store signature list and allow reverse query for location of corresponding data. We have the following definition:

Definition 3. (signature graph) [4] A signature graph G for a signature file $S = s_1.s_2... .s_n$, where $s_i \neq s_j$ for $i \neq j$ and $|s_k| = m$ for k = 1, ..., n, is a graph G= (V, E) such that

1. Each node v ∈ V is of the form (p, skip), where p is a

pointer to a signature s in S, and skip is a non-negative integer i. If i > 0, it tells that the ith bit of s_q will be checked when searching. If i= 0, s will be compared with s_q.

2. Let e= (u, v) ∈ E. Then, e is labeled with 0 or 1 and

skip(u) > 0. Let skip(u) = i. If e is labeled with 0 and i > 0, the ith bit of the signature pointed to by p(v) is 0. If e is labeled with 1 and i> 0, the ith bit of the signature pointed to by p(v) is 1. A node v with skip(u) = 0 does not have any children.

Example 10. Consider the signature file and signature graph:

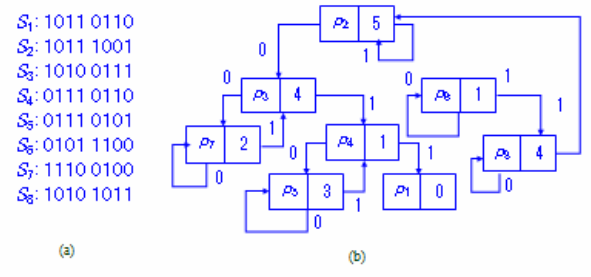

Figure 6. Signature file and signature graph

- *Algorithm 2 [4] signature-graphs-search*

Input: a query signature s_q ;
Output: set of signatures which survive the checking;
Method:

Step 1. Set ← ∅

Step 2. Push the root of the signature tree into stackp.

Step 3. If $stack_p$ is not empty, then v ← pop($stack_p$); else return(*S*).

Step 4. If *v* is not a market node and skip(*v*)≠ 0, *i* ← skip(*v*); mark *v*;

Step 5. Compare s_q with the signature pointed to by *p*(*v*).

(**p*(*v*) – pointer to signature*)

If s_q matches, then $S \leftarrow S \cup \{p(v)\}$.

Step 6. Go to(3)

3. PROPOSED INDEXING TECHNIQUE

3.1. Improved Algorithm for Signature Graph Search

In this part, we propose improvement of query algorithm on signature graph [4] which can be used to improve query time as follows:

Algorithm 3 signature-graphs-search

Input: a query signature Sq;

Output: set of signatures which survive the checking;

Method:

Step 1. Compute the Signature weight for the query signature.

Step 2. Set←Ø

Step 3. Push the root of the signature tree into $stack_p$.

Step 4. If $stack_p$ is not empty, v←pop($stack_p$); else return (Set).

Step 5. If v is not a marked node and skip ≠ 0, i←skip(v); mark v;

If S_q=0, push C_r and C_l into $stack_p$; (where C_r and C_l are v's right and left child, respectively) otherwise, put only C_r into $stack_p$.

Step 6. If signature weight is smaller than 50% then search the query signature in the signature nodes for the unset bits.

Step 7. Else search the query signature in the signature nodes for the set bits

Step 8. Compare S_q with the signature pointed by p(v).

(p(v) pointer to a signature)

If S_q matches, Set ← Set ∪ {p(v)}.

Step 9. Go to (3).

3.2. Time Complexity

From [4], query time complexity on signature graph is $O(N/2^l)$, where N is the number of signatures in the signature file and l is the number of bit 1 put in query signature S_q.
If query signature weight is higher than 50% then the number of bit 1 is larger than the number of bit 0 in S_q. Let k be the number of bit 0 put in query signature S_q, then we have $O(N/2^l) > O(N/2^k)$. Otherwise, $O(N/2^l) < O(N/2^k)$.
The above analysis shows that if signature weight is higher than 50%, comparison between query signature and signature of signature file will be based on bit 1. Otherwise, it will be based on bit 0, this way can improve query time on signature graph.

3.3. Clustering Algorithm

How is it if query returns zero? In this case, to find out closer match, Hierarchical Clustering is applied. In cluster,

similar objects are arranged together to create cluster. Because similar cluster objects will be suitable with any requirements (if any). This thing will increase search speed. This process included 2 step. First of all is cluster and the second is cluster searching. In hierarchical clustering, objects is linked with each other. In here, data structure of n-ary tree is used for creating cluster.

Algorithm 4. Clustering algorithm
Input: Condition collections.
Output: Qualified object is embedded in n-ary tree
Method:
Step 1. Creating new node and inserting into graph.
Step 2. Seeking object that being content with available condition and attaching new condition.
Step 3. Inserting new node with graph that being content with new condition.

For searching almost identical similarities, locating position of condition collections is provided in hierarchy system and seeking father node of it. For locating position of node, using Level Order Tree Traversal. Algorithm to find father node such as follows:

Algorithm 5. Parent node
Input: Object Condition collections.
Output: Parent node.
Method:
Step 1. Searching nodes are suitable with conditions.
Step 2. Returning qualified nodes to father node.

After gaining closest match, query is edited and information is retrieved. So therefore hierarchical tree helps for searching almost identical match.

4. Approach Combining Signature File Hierarchy and Signature Graph

4.1. Query Data Structure Model

To improve query time on databases, we need to describe data structure in a more simple way and build a corresponding data structure to reduce searching space during implementing query while ensuring query of necessary objects by using signature graph. From [4], to make query more optimized, we need to combine signature file hierarchy and signature graph, this issue has been proved to improve query time better. From algorithm 3, query time complexity on signature graph is smaller than query time complexity of algorithm 2. Thus, we still use signature file hierarchy as in [4], but replace algorithm 2 with algorithm 3 to improve query time better.

Base on theoretical basis and suggested algorithm, the paper proposes improved approach for query algorithm on signature graph combining signature file hierarchy as follows: (1) Each signature file is stored in signature graph structure to speed up signature file scanning; (2) All of signature files are organized in hierarchy to facilitate implementing step by step filter technique.

Example 11. Construction of signature graph is illustrated as below:

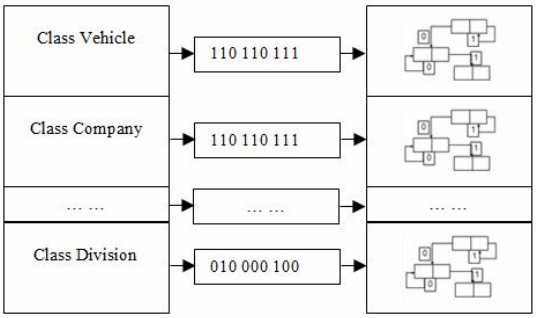

Figure 7. Construction signature graph

Example 12. Combination of signature file hierarchy and signature graph is illustrated as follow:

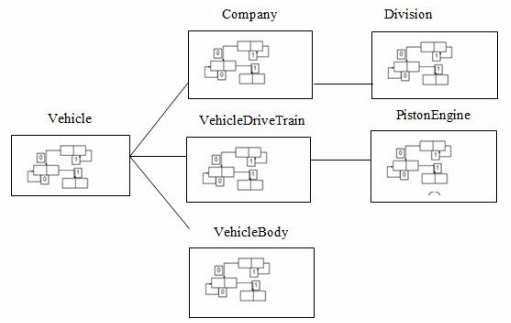

Figure 8. Signature file hierarchy and signature graph

Data structure is totally stored in the main memory, in this case, inserting or deleting a signature on a signature graph can be done easily. However, files are very large in databases, so database structure cannot store in the main memory but in the external memory. For object-oriented databases, they will be stored and implemented on the external memory. An object-oriented database has many classes, each class has many objects. There is a signature graph structure corresponding with each class, also each object will form an object signature. The entire object-oriented database is partitioned in a hash table structure including object's signatures to implement query process.

4.2. Object-Oriented Query Processing

To execute a query of an object in an object-oriented database, firstly we have to change an object-oriented database into data structure as above, we do:

Step 1. Attribute of the object is hashed into binary signatures and attributes which form object signatures.
Step 2. Object signatures in a same layer will form signature graphs.
Step 3. Create signature file hierarchy where each file is a signature graph.
After having data structure for query, we execute object query process on object-oriented databases as follow:
Step 1. Encode key words which need to be retrieved into binary signature.
Step 2. Execute key word signature query to determine classes which need to be searched.
Step 3. Execute key word signature query on signature graphs corresponding with determined classes.
Step 4. In case exact match is unavailable we change to step 5. Opposite, turning to step 6.
Step 5. Finding closest match.
Step 5.1. Using Clustering Algorithms on n-ary tree to create new requirement.
Step 5.2. Finding out information is suitable with new requirement.
Step 5.3. Seeking match of son node and returning corresponding father node.
Step 6. Updating information in database.

5. Conclusion

In this paper, we suggest to improve query algorithm on signature graph. Signature graph is created based on signature file for a layer and helping to improve effectively for searching on signature file. Plus, in case exact match is unavailable, n-ary tree is used for locating corresponding coincident position. Signature files are created as a hierarchical system according to nested structures of layers in Object-oriented database system helping to improve effectively for query time.

References

[1] S. Sung and J. Fu, (1996), Access Methods on Aggregation of Object-Oriented Database. *IEEE International Conference*, Vol (2), pp.977-982.
[2] Yangjun Chen, Yibin Chen (2006), On the Signature Tree Construction and Analysis, *IEEE Trans Knowl Data Eng*, 18(9), pp.1207-1224.
[3] Yangjun Chen (2004), Building Signature Trees into OODBs, *Journal of Information Science and Engineering*, 20(2), pp.275-304.
[4] Yangjun Chen, Yibin Chen (2004), Signature File Hierarchies and Signature Graphs: a New Index Method for Object-Oriented Databases, *Proceedings of the 2004 ACM symposium on Applied computing, Nicosia, Cyprus*, pp.724-728.
[5] D. Dervos, Y. Manolopulos and P. Linardis (1998), Comparison of signature file models with superimposed coding, *J. of Information Processing Letters 65*, pp.101 - 106.
[6] C. Faloutsos, (1985), Signature Files: Design and Performance Comparaison of Some Signature Extraction Methods, *ACM Sigmod Record*, Volume 14, Issue 4, pp. 63 – 82.
[7] D. L. Lee, Y. M. Kim, G. Patel, (1995), Efficient Signature File Methods for Text Retrieval, *IEEE Tran Knowl Data Eng*, 7(3), pp.423-435.
[8] W. C. Lee, D. L. Lee, (1992), Signature File Methods for Indexing Object-Oriented Database systems, *Proceedings of the 2nd International Computer Science Conference*, Hong Kong, pp.616-622.
[9] P. Mahatthanapiwat, (2010), Flexible Searching for Graph Aggregation Hierarchy, *Proceedings of the World Congress on Engineering*, London, UK, pp.405-409.
[10] E. Tousidoua, P. Bozanis, Y. Manolopoulos, (2002), Signature-based structures for objects with set-valued attributes, Elsevier Science, *Information Systems*, 27(2), pp.93-121.