Querying Object-Oriented Databases Based on Signature File Hierarchy and Signature Graph

Tran Minh Bao(✉) and Truong Cong Tuan

College of Science, Hue University,
77 Nguyen Hue Street, Hue City, Viet Nam
tmbaovn@gmail.com, tctuan_it_dept@yahoo.com

Abstract. Chen and his partners [2] proposed an approach which combines nested signature file hierarchy and signature graph as follow: (1) all files containing signatures are organized in a hierarchy for a quick filter of unsuitable data; (2) Each signature file is stored in a graph structure (called signature graph) to speed up signature scanning. This technique reduces significantly searching space, so it improves significantly query time complexity. In this paper, we improve query algorithm on signature graph based on the approach proposed by Chen and his partners, to improve query time on signature graph.

Keywords: Object-oriented query · Object signature · Signature file · Signature graph

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 [1–3, 8, 9].

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 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 an improved approach of algorithm on signature graph. Part 4 proposes an approach combining signature file hierarchy and signature graph. Finally, part 5 gives out a conclusion.

P.C. Vinh and V. Alagar (Eds.): ICCASA 2015, LNICST 165, pp. 298–308, 2016.
DOI: 10.1007/978-3-319-29236-6_29

2 Background

This part only presents some basic concepts related to object signatures, signature files. Further information can be found in [1, 2].

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 CSDL system, an entity displayed according to object type including methods and properties. Objects have similar methods and properties gathered in the same layer. If the *C* layer has a complex property with domain *C'*, so we shall create relation between *C* and *C'*. This relation is general relation. When using arrow to connect layers for displaying general relation, need to create general hierarchy for displaying nested structure of layers.

Example 3. This is an example about nested object hierarchy system illustrated like this:

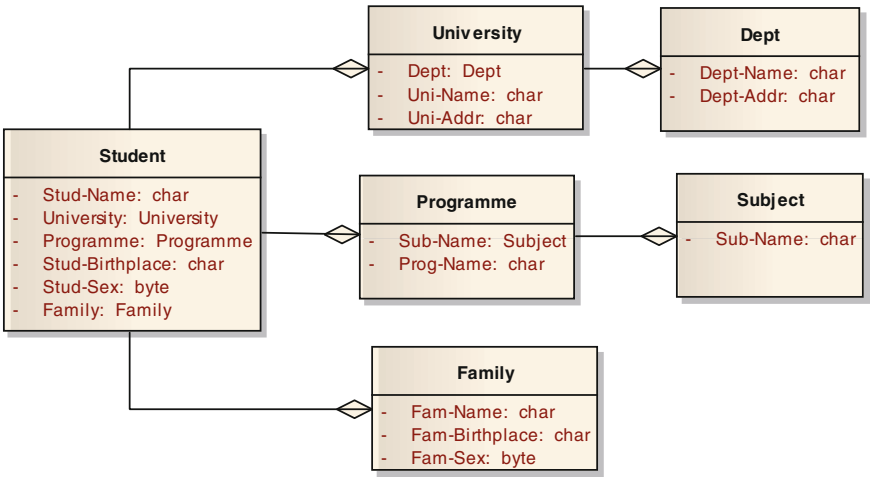

Fig. 1. An example of a nested object hierarchy

Object *o* referenced is a property of object *o'*, then object *o* considered as nested with each other in *o'*, and *o'* considered as 'father-object' of *o*.

In object-oriented CSDL, condition found in query collected in a collection of properties. This property is a nested property of target layers.

Example 4. The query "retrieve all students born in *Ben Tre* of dept *information technology*" can be expressed as:

```
Select Student
Where Student.Stud-Birthplace = "Ben Tre"
And   Student.University.Dept.Dept-Name   =   "information
technology"
```

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 Student and single out those who were born in *Ben Tre*. Then, the system retrieves the University objects referenced by the Student born in *Ben Tre* and checks the Dept-Name of the Dept. Finally, those Students born in *Ben Tre* by a University that has Dept *information technology* are returned.

2.5 Signature File Hierarchy and Query Algorithm

2.5.1 Signature File Hierarchy

Purpose of using signature file: remove unconditional objects, it means if we have a signature is not suitable with query signature so the object related with this signature surely ignored. So therefore we do not need to access to these objects.

Example 5. Signature and signature file hierarchy:

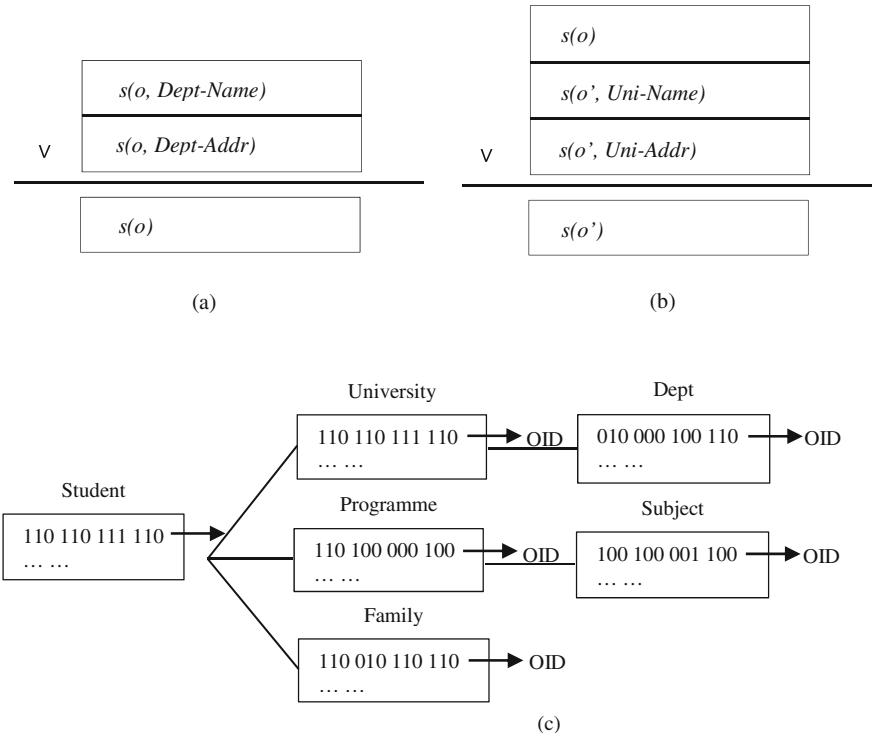

Fig. 2. Signature and signature file hierarchy

Considering Dept layer in *0* hierarchy of complex properties in Fig. 1. Signature of *o* object can be created by method in Fig. 2(a), each *s(o, x)* signature symbol created for property value *x* of *o* and *s(o)* signature symbol *o*. To layers of complex properties, signature of objects can be created with the same method, like layer of original properties. Difference: signature of complex property is signature of referenced object illustrated in Fig. 2(b). In Fig. 2(b), *o'* marked object of University layer. And *o* object of Dept layer is Dept's property value of *o'*. Hierarchy of signature file is used for creating databases displayed in Fig. 1 also illustrated in Fig. 2(c).

2.5.2 Query Algorithm Based on Signature File

We use query signature-tree to decrease searching-space. In this method, we need two *stack* structures to control prioritize scanning according to depth of tree structures: $stack_q$ to *Q(s, t)* and $stack_c$ to class hierarchy. In $stack_q$, each component is a signature, meanwhile in $stack_c$, each component is a collection of objects belong to the same layer can be approached by scanning class-hierarchy.

Algorithm 1. [4] 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.

This technique helps for optimization when we implement step (4). In this step, some objects selected by using corresponding signature in query signature tree. In step (5), referenced objects and son-node's signatures of query signature tree is added to $stack_c$ and $stack_q$. In step (7), we will conduct inspection on errors.

Example 6. Assuming we have a part of signature file hierarchy is created for a CSDL based on a diagram in Fig. 1 belongs to type described in Fig. 3:

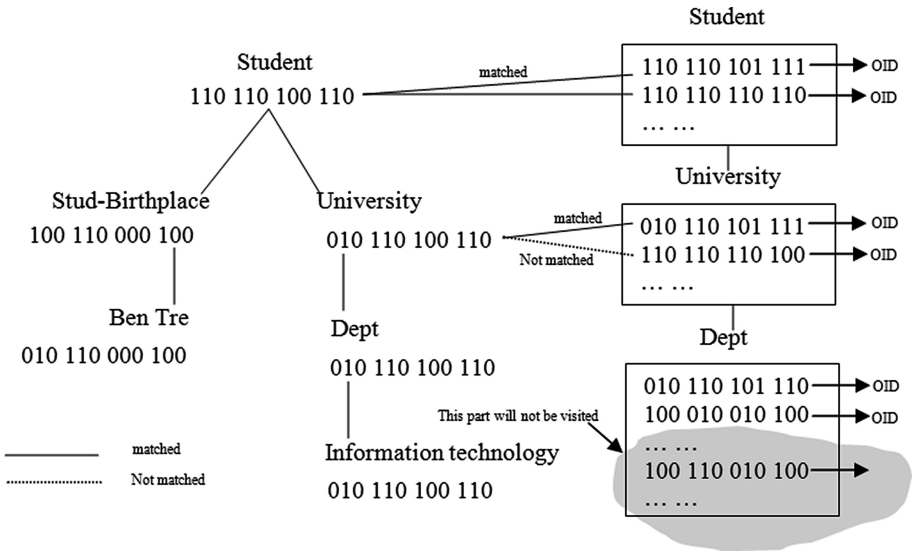

Fig. 3. Illustration of query evaluation

When the first signatures of signature-file for Student is suitable with signature in query signature tree, signatures are referenced by themselves in signature file for University need to have additional inspection. Assuming we have the first signature of University is referenced by the first signature in Student meanwhile the second signature in University is referenced by the second signature in Student. We can see that the second signature in University is not suitable with corresponding signature in query signature tree. Therefore, all signatures of Dept object is referenced by Dept object won't be inspected (watching grey illustration in Fig. 3). This method is optimal method when comparing with "searching from top to bottom" because in "searching from top to bottom" must inspect all Dept's object-signatures.

2.6 Signature Graph

2.6.1 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 7. 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:

Example 8. Consider the signature file and signature graph (Fig. 4):

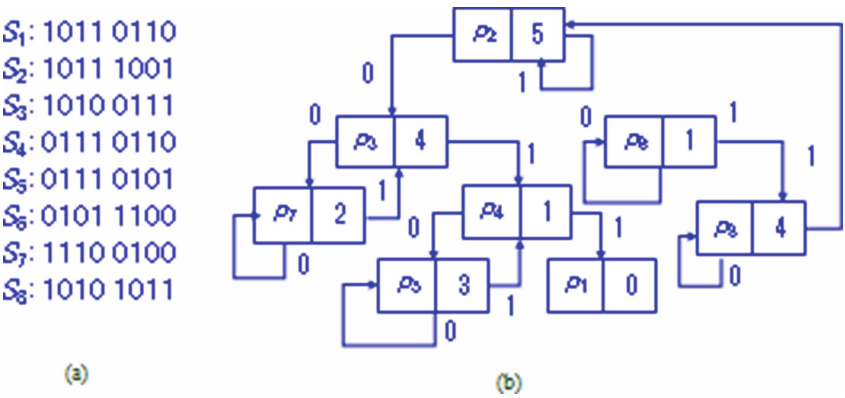

Fig. 4. Signature file and signature graph

3 Improved Algorithm

In this part, we propose improvement of query algorithm on signature graph [3] which can be used to improve query time as follows:

3.1 Improved Algorithm for Signature Graph Search

Algorithm 2 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 [3], 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.

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 [3], 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 [3], 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 9. Construction of signature graph is illustrated as below (Fig. 5):

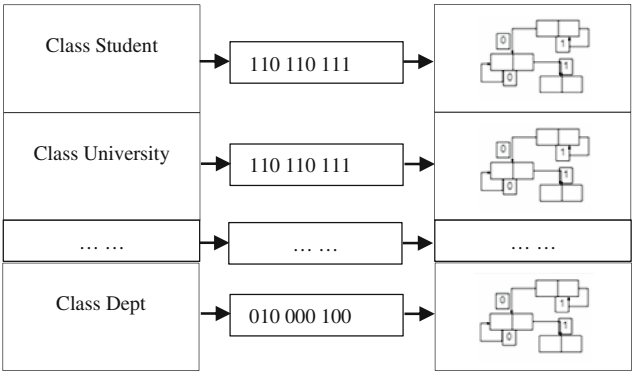

Fig. 5. Construction signature graph

Example 10. Combination of signature file hierarchy and signature graph is illustrated as follow (Fig. 6):

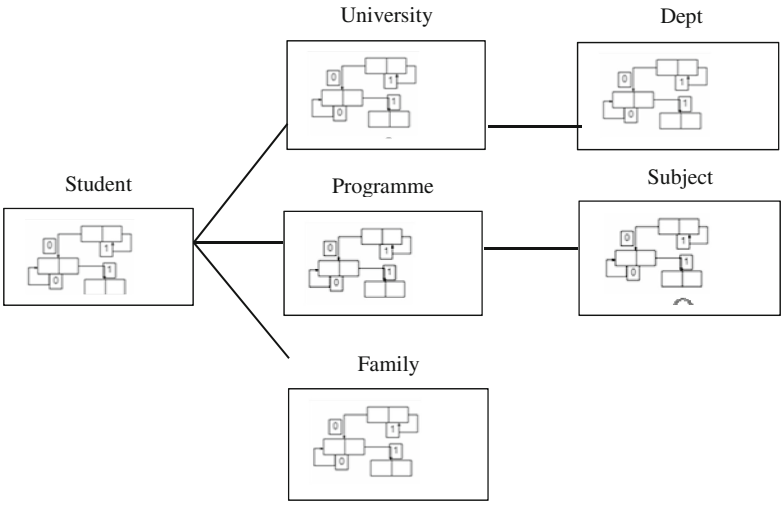

Fig. 6. Signature file hierarchy and signature graph

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.
```

5 Conclusion

In this paper, we propose a query algorithm improvement on signature graph. Signature graph structure is built on signature file for a class and help improve significantly signature file searching. Signature files are built up to a hierarchy with structure of nested classes in an object-oriented database to improve significantly query time.

References

1. Chen, Y., Chen, Y.: On the signature tree construction and analysis. IEEE Trans. Knowl. Data Eng. **18**(9), 1207–1224 (2006)
2. Chen, Y.: Building signature trees into OODBs. J. Inf. Sci. Eng. **20**(2), 275–304 (2004)
3. Chen, Y., Chen, Y.: Signature file hierarchies and signature graphs: a new index method for object-oriented databases. In: Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, pp. 724–728, 14–17 March, 2004
4. Dervos, D., Manolopulos, Y., Linardis, P.: Comparison of signature file models with superimposed coding. J. Inf. Proc. Lett. **65**, 101–106 (1998)
5. Faloutsos, C.: Signature files: design and performance comparaison of some signature extraction methods. ACM Sigmod Rec. **14**(4), 63–82 (1985)
6. Lee, D.L., Kim, Y.M., Patel, G.: Efficient signature file methods for text retrieval. IEEE Trans. Knowl. Data Eng. **7**(3), 423–435 (1995)
7. Lee, W.C., Lee, D.L.: Signature file methods for indexing object-oriented database systems. In: Proceedings of the 2nd International Computer Science Conference, Hong Kong, pp. 616–622 (1992)
8. Mahatthanapiwat, P.: Flexible searching for graph aggregation hierarchy. In: Proceedings of the World Congress on Engineering, London, UK, pp. 405–409, June 30–July 2, 2010
9. Tousidoua, E., Bozanis, P., Manolopoulos, Y.: Signature-based structures for objects with set-valued attributes. Elsevier Sci. Inf. Syst. **27**(2), 93–121 (2002)