Applying Signatures for Forward Traversal Query Processing in Object-Oriented Databases

Hwan-Seung Yong, Sukho Lee, Hyoung-Joo Kim
Dept. of Computer Engineering
Seoul National Univ., Seoul, 151-742, KOREA
E-mail: {hsyong, shlee}@krsnucc1.bitnet, hjk@im4u.snu.ac.kr

Abstract

Forward traversal methods are used to process queries having nested predicates in object-oriented databases. To expedite the forward traversal, a signature replication technique is proposed. Object signature is a signature formed by values of all atomic attributes defined in the object. When an object refers to other objects through its attribute, the object signature of the referred object is stored into the referring object. Using object signatures, nested predicates can be checked without inspecting referred objects.

1 Introduction

In object-oriented databases, the attribute of an object can have the object-identifier(OID) of another object as a value. Through this OID, objects can refer to other objects. That is each object of a class can refer to an object or objects of other class by defining the referred class as the domain of the attribute of the referring class. We follow the lead of [1] in defining terms for this paper.

Those classes related with their attributes form a hierarchical structure called *class-attribute hierarchy* [1]. Figure 1 is an example class-attribute hierarchy; the hierarchy is rooted at the class Student, and the '*' symbol next to an attribute indicates that the attribute is multi-valued. An attribute of any class in a class-attribute hierarchy is logically an attribute of the root of the hierarchy, that is, the attribute is a *nested attribute* of the root class [1]. The attribute name of the class Department is a nested attribute example of the class Student represented by Student.dept.name.

A predicate that has a *nested attribute* is called a *nested predicate*[1]. A predicate such as Student.dept.name = 'Computer' is an example of a nested predicate of the class Student. To evaluate queries with nested predicates, three traversals(called *forward traversal*, *backward traversal* and *mixed traversal*) are proposed in [2].

Two techniques have been proposed for fast evaluations of queries having nested predicates [1, 3]. One is to use *nested index* [1] to index nested attributes, and the other is to use *field replication* techniques [3] to avoid forward traversals.

The field replication technique copies attribute values of the frequently referred object and stores them into the referring object. As such, during forward traversal process, there is no need to access the referred object to retrieve its attribute values. However the field replication technique might can cause high storage cost and data inconsitency problem.

Indexing techniques can efficiently support backward traversals. But this requires high storage and maintenance costs. These overheads may constrain to limit indices for multiple attributes of the classes. Therefore, on the whole, indices are maintained only for important attributes.

In this paper, as an another approach, signatures [4] for each object (called object signatures) are generated from its attribute values and stored into the referring objects to support forward traversals. Using this technique, nested predicates can be checked by using these signatures without accessing referred objects. Only when the predicates are successfully matched with the signature will the forward traversals continue to read values of the referred objects. Also, the characteristic that the signature file has low storage cost [5], will allow all attribute values of objects to be included to make the signatures. This makes it possible to evaluate any predicates on the classes. In the case where nested indices are also defined, it is possible to evaluate queries using mixed traversals which have both the nested index and the signatures.

The organization of this paper is as follows. In section 2, forward traversals for query processing and signature approaches are described. In section 3, a

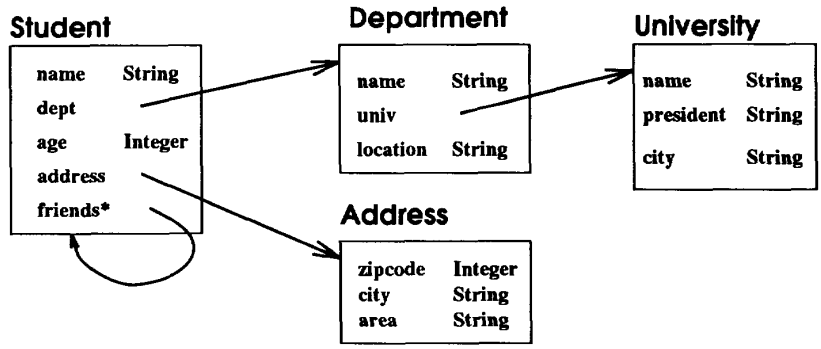


Figure 1: Class-attribute hierarchy

new forward traversal technique using signatures is described. Section 4 handles the cost evaluation for the proposed technique. Then we conclude in section 5.

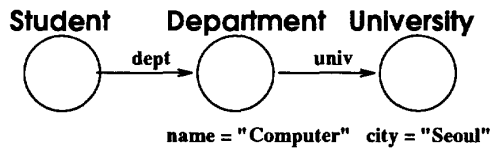


Figure 2: Query graph for Q1

2 Forward traversals and signatures

2.1 Forward traversals

Three ways to visit class nodes in a query graph which is formed by a query on object-oriented database are proposed [2]. Unless indices are provided, forward traversal could be normally used for processing queries. Below we briefly overview the three traversals.

- Forward traversal: Visit the root class node first and visit other classes in any depth-first ordering.
- Reverse traversal: Visit leaf class nodes first and proceeds in a bottom-up route along the graph.
- Mixed traversal: Visit class nodes by combination of the two traversal methods above.

To explain the forward traversal, consider a query Q1 on the class-attribute hierarchy example in Figure 1 and its query graph given in Figure 2.

> Q1: Retrieve student information whose department is 'Computer' and university is located at 'Seoul'.

If no indices are defined for the two nested attributes, Student.dept.name and Student.dept.univ.city, given in Q1, the query can be evaluated using the forward traversal. In this case, all objects in class Student should be retrieved. And for each object in Student, the referred object in class Department through the attribute dept of Student will be retrieved to check that its value of the attribute name is 'Computer'. When this condition does not satisfy, the same procedure will be applied on the other Student objects. But when the condition is satisfied, referred objects in the class University through the attribute univ of Department will be retrieved. And for the retrieved University object, the predicate on the University will be checked to see whether its value of the attribute city is 'Seoul'.

2.2 Signature

Signature file indexing scheme, proposed for multi-key indexing, is now extending its application areas to text retrievals [4] and multi-media data retrievals [6, 7].

There are several methods for generating signatures from data [8], but here we only consider the superimposed coding method. Signature generation procedure through superimposed coding from data D={G.D.Hong, 25, Seoul} for the signature of D, S_D, is shown in Table 1. In this process, we first hash each value using a function having two parameters b and k, which represents hashing size of bits and number of bits to be set as '1' respectively. The signature is then simply derived from ORing all these hashing results.

Using this signature, we can check whether the

Value	Hashing result
G.D.Hong	1001 0000 1000 0001
25	1000 1100 0000 0100
Seoul	1000 0000 1100 1000
signature S_D	1001 1100 1100 1101

Table 1: Signature generation example(in case b=16, k=4)

given value is in a signature or not. This checking procedure (called *signature matching*) selects candidate signatures to be examined. A signature S is qualified if and only if, for all bit-1 positions in the query signature, the corresponding bit positions in S are also set to 1.

Let '$\cap$' be the bitwise 'AND' operator and let S_Q be the signature of query Q. Then matching procedure can be formally represented as below.

$$(S_D \cap S_Q) = S_Q$$

Suppose multiple attribute values like "Seoul" and 25 are given to query signatures as conjunctive (logical AND) conditions. If the hashing result of 26 is '1000 0100 1000 1000', then this also matches successfully with S_D. We call this a *false drop*.

Signature files require low storage overhead and can provide multi-attribute indexing. Recently, signature file indexing is proposed for object-oriented databases [9]. But when it is used in indexing, retrieval performance degrades as the number of records increases.

3 Forward traversal technique using signatures

3.1 OID table and Object signatures

Object signature is a signature formed by values of all atomic attributes defined in the object. Two methods are possible to represent OID. First, physical address of an object can be used as OID directly. Second, OID can be represented using logical addresses (called surrogates) with a mapping table (we call OID table) which maps each surrogate to its physical address. The first method generally provides good retrieval performance. However, as objects relocate, retrieval performance becomes worse and OID may not be unique unless other technique is used. The second one can make objects independent on storage structures. But whenever objects are accessed, OID table search is necessary [10]. In this paper, we assume that the OID table is used to map a surrogate to a physical address. Object retrieval procedure using the OID table consists of three steps shown in Figure 3 (a) when an OID and some predicates are given.

1. Retrieve a physical address of the object from the OID table
2. Retrieve values of the object using the physical address
3. Check predicates from the values retrieved from the above step

Therefore, at least two disk I/Os are required.

But when object signatures are stored in the OID table with its OID, retrieval procedure with predicates consists of the following steps. The precedure is also shown in Figure 3 (b). Throughout the paper, an object signature of OID is represented as S(OID) for simplicity.

1. Retrieve a physical address and the object signature from the OID table
2. Check predicates from the object signature
3. Only when the above checking satisfies, they can retrieve values of the object using the physical address

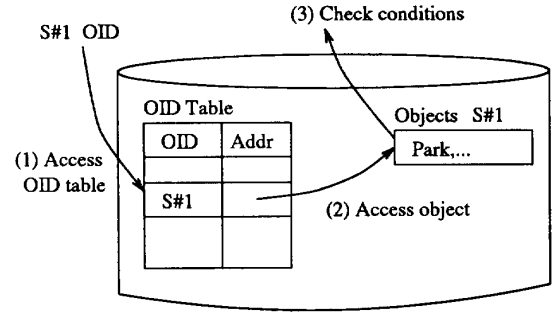
Therefore, using the object signatures stored in the OID table, query predicates on the object can be checked without retrieving actual values of the object.

3.2 Reference relationships and object signatures

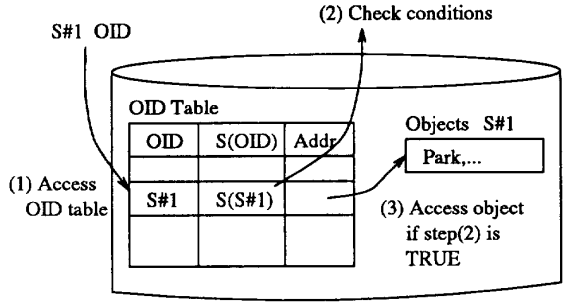
The fact that an object A refers to an object B through its attribute means that using this attribute, values of the object B can be retrieved from the object A during operations on the object A. Accessing referred objects requires another disk access. But if we know the values of referred object, when we are accessing referring object during evaluation of nested predicates, it is not necessary to access the referred object.

The field replication technique [3] stores frequently accessed attribute values of the referred object into the referring object.

Unlike the field replication technique, the object signatures (not the attribute values) are replicated in our approach. That is, when any object has OIDs of another object as its attribute values, not only their OIDs but also their object signatures are stored.

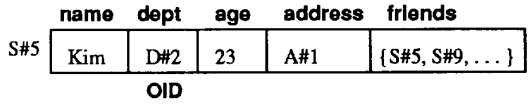


(a) Conventional retrieval procedure.

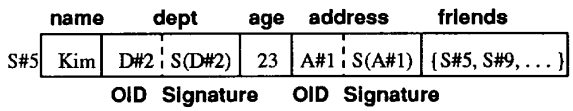


(b) Retrievals using signatures

Figure 3: Object retrieval procedure using OID



(a) Conventional object reference structure



(b) Reference structure of OID and object signature

Figure 4: Reference structure using OID

Figure 4 shows a reference structure that shows the values of an object S#5 in the class Student with reference relationships.

3.3 Forward traversal algorithms using signatures.

To explain query processing procedures using forward traversals: suppose an object A refers to an object B and nested predicates are given on the class to which the object B belongs. Then the procedure according to previous forward traversal is as follows.

1. Retrieve the referring object A.
2. Retrieve the object B which the object A refers to through its attribute.
3. Check predicates using values of the object B.

In contrast, forward traversal steps using signature replication are as follows.

1. Retrieve referring object A
2. Check predicates using signature of the referred object in object A
3. Only when above checking satisfies, retrieve the referred object B

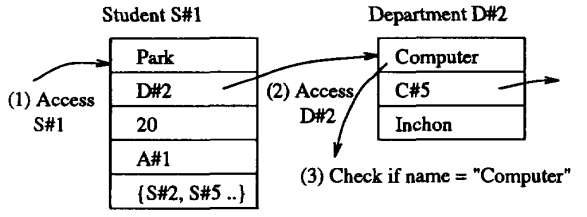
Consider an example query Q2 on classes in Figure 1.

> Q2: Retrieve students in 'Computer' department.

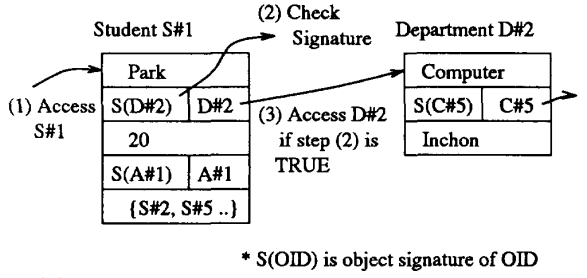
Suppose an object S#1 in class Student refers to an object D#2 in class Dept through an attribute dept.

Then in the previous forward traversal procedure, as shown in Figure 5 (a), we need to also retrieve the D#2 object after retrieving S#1 object. Using values of D#2, we can check whether the value of an attribute name is "Computer" or not. But in the forward traversal procedures using signatures, as shown in Figure 5 (b), we can check the predicates using the object signature S(D#2) which is also stored with OID D#2 in the object S#1 after retrieving S#1 object. Only when the signature successfully matches, retrieval of the object D#2 is performed.

In the inverted indexing scheme, we might have to search many indices for complex queries. But in the signature replication technique, the filtering effect of signatures is even more increasing for complex queries. As a consequence, the retrieval performance of the signature replication method is superior to the inverted indexing scheme.



(a) Conventional forward traversal procedure.



(b) Forward traversal procedure using signatures.

Figure 5: Comparison of forward traversals

3.4 Management of object signatures

3.4.1 Creation of object signatures

When an object is created, the corresponding object signature is stored in the OID table. When one object is referring to other object through its attribute, the object signature of the referred object is stored with its OID into the referring object. Therefore, when one object refers to multiple objects through different attributes, then the same number of object signatures need to be stored in the referring object.

3.4.2 Size of object signatures

The size b of an object signature varies according to which class the object belongs to. Generally, the size b of a signature is determined by the number of attribute values which make the signature and by the false drop probability. The formula to calculate b is given as follows[11]: let *nattr* be the number of attributes and F_d be the false drop probability.

$$b = (1/log2)^2 \cdot nattr \cdot log(1/F_d) \qquad (1)$$

3.4.3 Updates of object signatures

Object signatures need to be updated whenever any value of the object changes. Therefore, all object signatures stored in the referring object should also be changed correspondingly. To update them with as low cost as possible, it is necessary to know which objects are referring to the referred object.

If the system supports above mechanisms by keeping all reverse pointers, it is simple to update the object signatures. Otherwise, we can use *link object* as in the field replication [3].

But in the case where only insertions and deletions of objects are allowed and the update of values are prohibited, it becomes unnecessary to update object signatures discussed above.

3.5 Mixed traversals with nested indices

Signature replication techniques can evaluate queries even more efficiently if the *nested indices* [1] are also defined. The nested indices can support backward traversals while signature replications support forward traversals. As a result, we can utilize both techniques.

Query Q1 has two nested predicates of attributes, Student.dept.name and Student.dept.univ.city. If one index is readily defined in one of the two attributes, we can process the query more efficiently by using mixed traversal methods. Therefore, when any index has been defined, then first use the index and later the signature replication technique.

Figure 6 generalizes mixed query processing procedures for different types of queries by showing possible query graphs and procedures. In the figure, A,B,C and D represent class names and we assume at least one predicate is given for each class. When nested indices are defined, we also assume that the indices are defined for the attributes of given predicates.

Type IV represents the case where multiple predicates are given on class B and one attribute is defined to have a nested index.

As we can see in the Figure 6, the basic rule of the processing procedure is to first apply nested indices when any index is provided. From the filtered objects after applying nested indices, we can process the query using signature based forward traversals.

For example, if a nested index on Student.dept.univ.city is defined, then we can select objects in the class Student using this index. For each objects selected above, we can select Department objects by the forward traversal algorithms using signatures. Query Q3 shows the case of two predicates

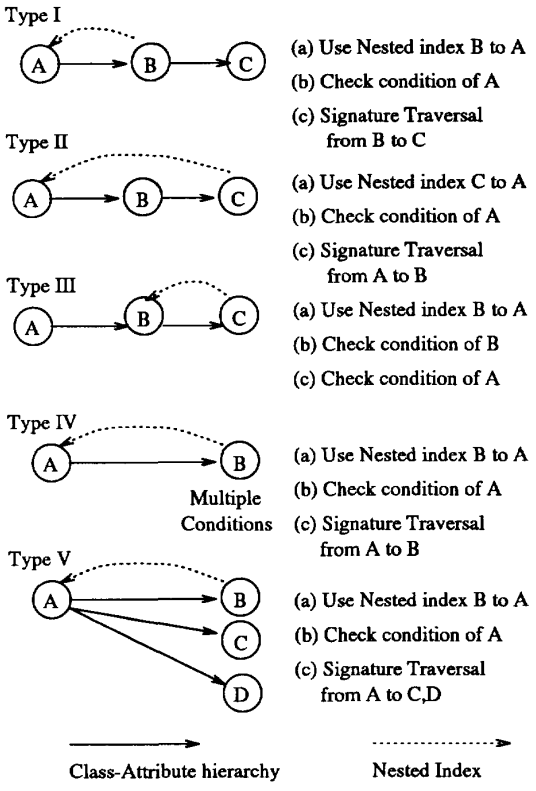


Figure 6: Mixed query processing using nested index and signature replication

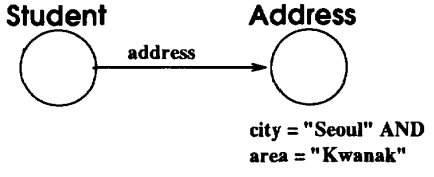


Figure 7: Query graph for Q3

on the class Address.

Q3: Retrieve students who live at Kwanak-Ku in Seoul.

Query Q3 has two nested predicates of attributes, Student.address.city and Student.address.area. When one of the two index is already defined (Student.address.city), we can retrieve Student objects using this index, and then use the forward traversal technique using signature replications from the retrieved objects to check another predicate on the class Address.

3.6 Set-valued attributes

An attribute can be defined to have a set of OIDs like the attribute friends of the class Student in the example classes of Figure 1. Signature replication can also be applied to this kind of attributes. For this purpose, object signatures should be stored for each OIDs.

For example, if one instance of the attribute friends = {S#1, S#2, S#5, S#10}, then its signature replicated instance also becomes friends = {(S#1, S(S#1)), (S#2, S(S#2)), (S#5, S(S#5)),(S#10, S(S#10))}.

Therefore, to process a query like "Retrieve students whose friend name is 'G.D.Hong"', we can check given predicates for object signatures in the attribute friends using the forward traversal algorithms.

4 Performance evaluation

In order to evaluate the performance of our approach, let us consider a general query having the query graph in Figure 8. For each class $C_i (1 \leq i \leq n-1)$, domain of its attribute A_i is defined as the class C_{i+1}. Number of n predicates $Pred_i (1 \leq i \leq n)$ is also given on the attributes $A_i' (A_i' \neq A_i)$ of the class C_i.

Parameters used for the evaluation are as follows.

- N_i: Number of objects in the class C_i
- M_i: Number of objects that satisfy a predicate $Pred_i$ in the class C_i
- P_i: Probability that an object in the class C_i satisfies a predicate $Pred_i$, $P_i = M_i/N_i$
- V_i: Number of objects in the class C_i that must be retrieved to process the given query
- F_d: False drop probability of signatures

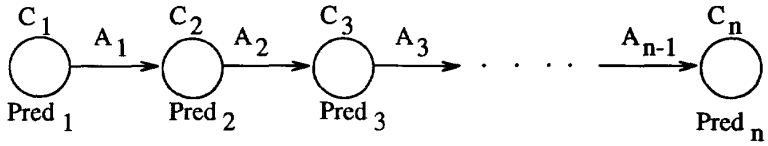


Figure 8: A general query graph

For performance evaluation, we only consider the number of objects that need to be retrieved during query processing. In the following subsection, we compare the signature replication technique with the nested loop technique using the forward traversal method.

4.1 Nested loop techniques

Number of objects in the class C_i that need to be retrieved, $V_i (2 \leq i \leq n)$, can be calculated as follows.

$$\begin{aligned} V_i &= (\text{Number of objects retrieved in the class } C_{i-1}) \\ &\quad \cdot (\text{Probability that the object in the class } C_{i-1} \\ &\quad \text{satisfies } Pred_{i-1}) \\ &= V_{i-1} \cdot P_{i-1} \end{aligned}$$

V_1 is equal to the number of objects of the class C_1 (N_1).

Therefore, the total number of objects retrieved, V, will be as follows.

$$\begin{aligned} V &= V_1 + V_2 + V_3 + \ldots + V_n \\ &= V_1 + V_1P_1 + V_1P_1P_2 + \ldots + V_1P_1P_2 \cdots P_{n-1} \\ &= V_1(1 + P_1 + P_1P_2 + P_1P_2P_3 + \ldots \\ &\quad + P_1P_2 \cdots P_{n-1}) \\ &= V_1(1 + \sum_{i=1}^{n-1} \prod_{j=1}^{i} P_j) \end{aligned}$$

4.2 Signature replication techniques

Number of objects in the class C_i that should be retrieved, $V_i (3 \leq i \leq n)$ can be calculated as follows.

$$\begin{aligned} V_i &= (\text{Number of objects retrieved in the class } C_{i-1} \\ &\quad \cdot \text{ Probability that the object in the class } C_{i-1} \\ &\quad \text{satisfies } Pred_{i-1} \\ &\quad - \text{ Number of objects retrieved by false drops} \\ &\quad \text{in class } C_{i-1}) \\ &\quad \cdot (\text{Probability that the object in the class } C_i \\ &\quad \text{satisfies } Pred_i \text{ using object signatures} \\ &\quad + \text{ False drop probability}) \\ &= V_{i-1} \cdot (P_{i-1} - (1 - P_{i-1})F_d) \cdot (P_i + (1 - P_i)F_d) \end{aligned}$$

V_1 is equal to N_1 and V_2 is calculated by $V_1P_1(P_2 + (1 - P_2)F_d)$ because there are no false drop objects in V_1.

Figure 9 shows the comparisons of the two techniques when sample data is given. Suppose three classes $C_i (1 \leq i \leq 3)$ are given based on the example query graph in Figure 8 and for each class, P_i is equal to 10%. The number of objects in the class $C_i (2 \leq i \leq 3)$ that should be accessed is shown in Figure 9 with varied F_d and N_i. We do not count the number of objects in the class C_1 (V_1) because V_1 is equal to N_1 which is independent on the selected techniques.

The figure shows that the signature replication technique requires fewer objects that should be retrieved than the nested loop technique. And there is a slight difference when the F_d is changed.

There is also a difference in the retrieval procedure of objects in the class C_1 between the two techniques.

The signature replication technique can check predicates using object signatures in the OID table, and therefore, requires fewer disk I/Os than nested loop techniques as explained in the subsection 3.1.

5 Summary

In this paper, the forward traversal method using signatures was proposed. A signature made from the attribute values of an object is stored into the OID table when the object is created. When any object refers to other object through its attribute, the object signature of the referred object is stored into the referring object. Using this technique, nested predicates can be checked on the referring object using stored object signature without accessing the referred object.

In addition, only by keeping object signatures in the OID table, physical object accesses after OID table accesses can be omitted.

In this paper, we have only considered forward traversals for direct reference relationships. Replication technique of object signatures for indirect referencing objects is at present being considered as further research. We also plan to consider values of unfor-

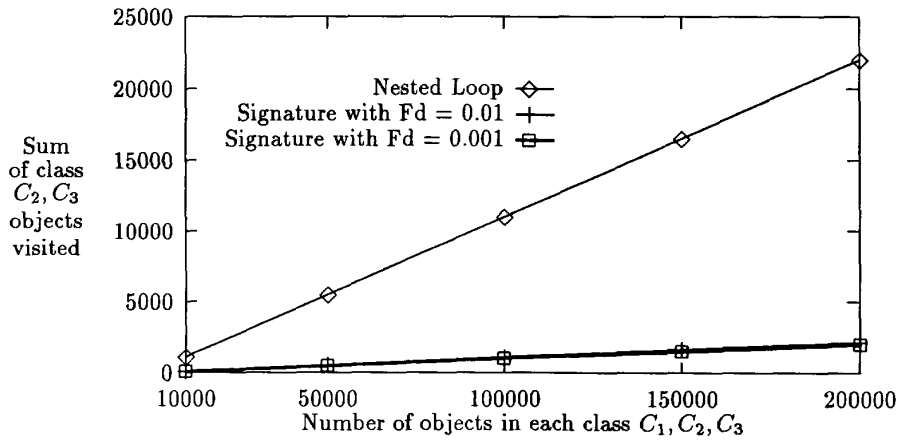


Figure 9: Performance comparison of forward traversals, in case $P_i = 0.1(1 \leq i \leq 3)$

matted objects such as multi-media data within the proposed framework.

Acknowledgement

We wish to thank anonymous referees for their helpful comments.

References

[1] E. Bertino and W. Kim, "Indexing Technique for Queries on Nested Objects," *IEEE Trans. on Knowledge and Data Eng.*, vol. 1, pp. 196–214, March 1989.

[2] K.-C. Kim, W. Kim, D. Woelk, and A. Dale, "Acyclic Query Processing in Object-Oriented Databases," in *Proc. Entity-Relationship Conference*, Nov. 1988.

[3] E. Shekita and M. Carey, "Performance Enhancement Through Replication in an Object-Oriented DBMS," in *Proc. ACM SIGMOD*, June 1989.

[4] C. Faloutsos, "Access Methods for Text," *ACM Computing Surveys*, vol. 17, pp. 49–74, March 1985.

[5] S. Christodoulakis and C. Faloutsos, "Design Considerations for a Message File Server," *IEEE Software Engineering*, vol. 10, pp. 201–210, March 1984.

[6] P. Zezula *et al.*, "Dynamic Partitioning of Signature Files," *ACM Trans. on Information Systems*, vol. 9, no. 4, pp. 336–369, 1991.

[7] F. Rabitti and P. Zezula, "A Dynamic Signature Technique for Multimedia Databases," in *Proc. of ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 193–210, Sept. 1990.

[8] C. Faloutsos, "Signature Files: Design and Performance comparison of some signature extraction methods," in *ACM SIGMOD Conference*, pp. 63–82, 1985.

[9] W. Lee and D. Lee, "Signature File Methods for Indexing Object-Oriented Database Systems," in *Proc. of Int'l Computer Science Conference(ICSC)*, pp. 616–622, 1992.

[10] R. Cattel, *Object Data Management: Object-oriented and Relational Database Systems*, pp. 151–152. Addison-Wesley Publishing Company, 1991.

[11] R. Sacks-Davis and K. Ramamohanarao, "A Two level superimposed coding scheme for partial-match retrieval," *Information Systems*, vol. 8, no. 4, pp. 273–280, 1983.