Flexible Searching for Graph Aggregation Hierarchy

Pichayotai Mahatthanapiwat

***Abstract*— Signature technique for multi-key indexing is proposed for flexible searching in the area of database. In this paper, the signature technique called the Join Signature file is presented to cover graph aggregation hierarchy. The structure of Join Signature is flexible enough so that the predicate can occur at any classes in the hierarchy and the target can be anywhere in the graph aggregation hierarchy. The retrieval performance when compared with the forward and reverse traversal technique is discussed and the cost models in terms of the storage cost and the retrieval cost are then formulated using the structure explained in the research.**

***Index Terms*—Join Signature File, Graph Aggregation Hierarchy, Retrieval and Searching, Object-Oriented Database.**

I. Introduction

Object-oriented databases provide data modeling mechanisms to support applications such as Computer Aided Design (CAD), Computer Aided Manufacturing (CAM) and Geographical Information Systems (GIS). The characteristics of object-oriented databases and their data model are different from those of relational databases. From the principle of the object model, a class is a template to create objects. Furthermore, a class allows its attribute to have complex data while the attribute of the relational model is limited to primitive data.

An object consists of its state and behavior. State of the object is its attribute value that can be primitive or complex. The complex attribute stores unique object identifier (OID) of other object. Behavior is a method or predefined procedure that operate on the state of the object.

Due to object model, the complex attribute A of class C can contain OID of object in domain class C' so aggregation relationship can be established between C and C'.

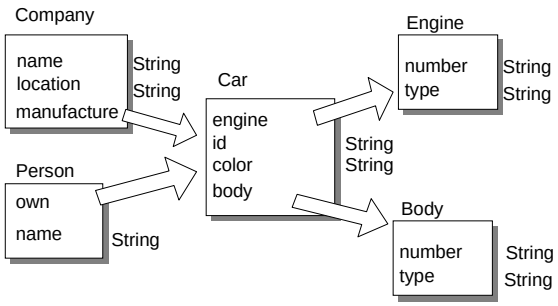

Fig. 1. Aggregation Hierarchy

Fig. 1 is an example of graph aggregation hierarchy, which consists of five classes. *Company* is an ancestor class and has two simple attributes; *name, location* and one complex attribute; *manufacture*. The domain class of attribute *manufacture* is *Car*, so OID of object in the class *Car* is stored in the *manufacture* attribute. The *Car* class has two parent classes, *Company* and *Person*. It can also refer to the *Engine* class and the *Body* class. Therefore, the *Car* class is an intermediate class. The *Engine* class does not refer to any classes, so it is the final class in the aggregation hierarchy.

Forward traversal is use to access objects by traversing from an ancestor class to the nested classes by using the pointer, which is the OID of its child class. Therefore, an object of the *Company* class can refer to its nested attribute *number* of the *Engine* class by traversing from the corresponding objects of the *Company* class, the *Car* class and the *Engine* class respectively. Conversely, traversing from a child class to its parent class is called reverse traversal. When there is a query, the class that the predicate is involved is called the predicate class and the class of the target objects is called the target class.

So far, many indexing techniques have been proposed getting rid of cost of traversal. Most of them dealt with indexing on nested attribute. [1] proposed nested index linking an object from the final class to objects in the ancestor class along the path. Path index [1] used the similar technique and stored additional OIDs of objects on the path. Indexing techniques used in both aggregation hierarchy and inheritance hierarchy are proposed in [2][3][4].

Queries that have predicate on the final class can get benefit from indexing techniques. However, queries with predicate on the ancestor class can hardly get the benefit. [5] proposed the technique of direct link of a path between object in the ancestor class and object in the final class.

If all classes have equal opportunity to be the predicate class, the techniques mentioned above are inapplicable in this situation. [7][8] proposed the technique called Join Signature to cope with this problem. Join Signature for tree aggregation hierarchy is presented in [9]. It is absolutely that indexing techniques require high storage and maintenance cost and may constrain the indices for multiple attributes. Signature technique will be the alternative approach for searching the target objects for classes in the graph aggregation hierarchy. Signature technique will be more flexible because we cannot always predict which key attribute will be used to access the database.

The organization of this paper is as follows. The brief of the signature file technique will be described in section 2. Section 3 describes joining of classes in the graph aggregation hierarchy. Structure and retrieval operation of the Join Signature File for graph aggregation hierarchy is then presented in section 4 and 5 respectively. Section 6 proposes storage overhead and the retrieval cost. Finally, section 7 summarizes the paper.

Pichayotai.Mahatthanapiwat is with School of Computer Engineering, Suranaree University of Technology, Nakhonratchasima Thailand 30000 (e:mail: pmh@sut.ac.th)

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

II. Signature File Technique

Let us review the concepts of the signature file technique used for object-oriented databases. A signature is an abstraction of the information stored in the attributes of an object [6]. Signature is created by superimposing bit string generated from the attribute values of the object being represented as shown in table 1.

Table I. Object Signature Generation

Value	**Hashing result**
Honda Auto	0101 0001 0010 0001
Bangkok	1010 0100 0001 1000
Object Signature	1111 0101 0011 1001

Table 1 shows the signature generation of an object in the Company class having two simple attributes, *name* and *location*. Signatures of simple attributes are obtained by hashing their attribute values into a bit string. An object signature is formed by superimposing or ORing all these hashing results. Object signatures are stored sequentially in the signature file. Using the signature, we can check if the given value is in a signature. A query specifying certain values to be searched for is transformed into a query signature by using the same hashing function as we used in generating bit string for attribute value. Then, we compare query signature with each object signature in the signature file. Object signature is qualified if and only if, for all bit-1 positions in the query signature, the corresponding bit position of object signature are also set to 1 ($S_Q \wedge S_O = S_Q$), where S_Q = query signature, S_O = object signature. However, after signature matching, we still have to examine the object in order to eliminate the false drop; signature matching but the object does not match the search criteria

III. Joining of Classes in Graph Aggregation Hierarchy

The graph aggregation hierarchy is shown in Fig. 2.

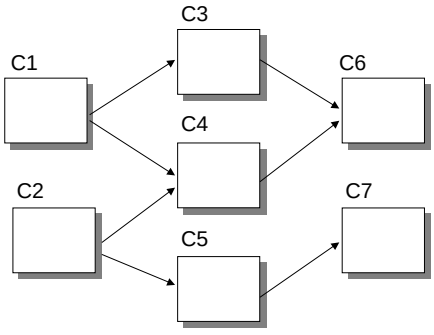

Fig. 2. Graph Aggregation Hierarchy

From the Fig. 2 above, C1 and C2 are ancestor classes whereas C6 and C7 are final classes. C3, C4 and C5 are linked and link to other classes so they are intermediate classes. Any objects in C1 can link directly and indirectly to the remaining classes in the graph aggregation hierarchy. Restructure of joining of classes in graph aggregation hierarchy is presented in Fig. 3.

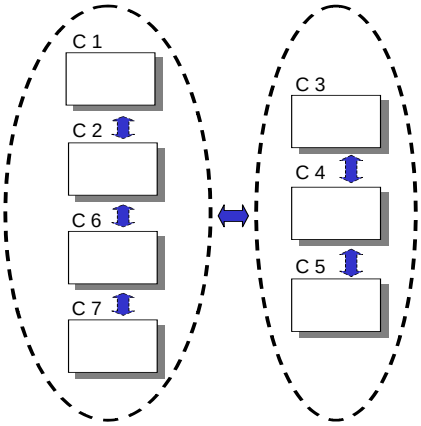

Fig. 3. Joining of Classes

There are 2 groups of joining between classes in Fig. 3 as follows

1. The joining of ancestor classes and the final classes (AF group).
2. The joining of intermediate classes (I group).

Then the AF group and I group are joined together.

IV. Structure of Join Signature File

Given the graph aggregation hierarchy of classes, We can get joining of classes and object relationship. The information of joining objects for joining classes will be described as follows.

For a graph aggregation hierarchy if O1 is an object in class C1 and O1 links to O2, O3, O4, O5, O6 and O7 of class C2 to class C7 whether it links directly or indirectly. The information of O1 to O7 will be kept together as OID relationship. The structure of OID relationship is shown in Fig. 4.

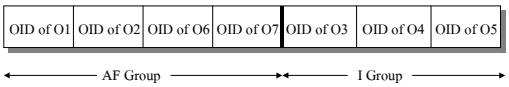

Fig. 4. The Structure of OID Relationship

The Join Signature for each joining group is calculated as follows

1. The Join Signature is obtained by superimposing the object signatures for objects relationship in the group.
2. The signature of an object is generated by superimposing the signatures of all of its simple attributes.
3. The signature of a simple attribute is obtained by hashing on the attribute value.

The structure of Join Signature for I group is shown in Fig. 5.

I Join Signature	OID of O3	OID of O4	OID of O5

Fig. 5. The Structure of an Entry of I Join Signature

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

The AF Join Signature will be obtained using the same method as of I Join Signature except using objects signatures of the AF group.

The Join Signature is generated by superimposing the object signatures of objects in the relationship for each group. It is generated as shown in Fig. 6. Entries of Join Signature will be stored in the file called the Join Signature file.

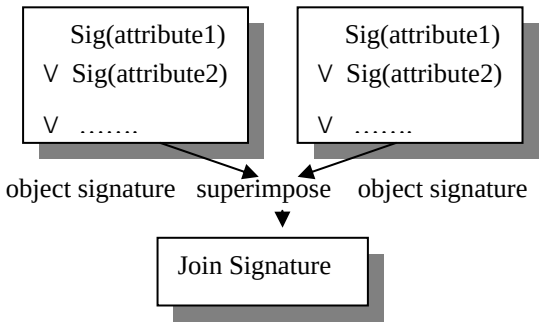

Fig. 6. The Join Signature

V. Retrieval Operation

An example of an entry in the Join Signature file will be given to describe the retrieval operation for the Join Signature. Fig. 7 shows an example of objects in an aggregation hierarchy.

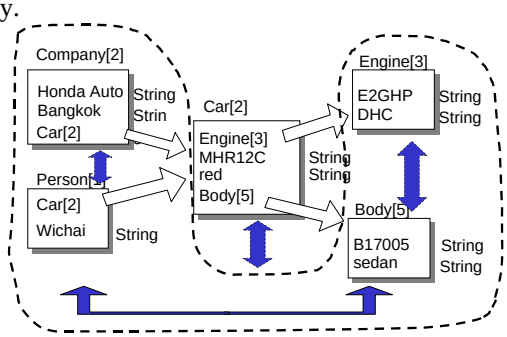

Fig. 7. An Example of Objects in Graph Aggregation Hierarchy

From Fig. 7, Company[2] represents OID of 2nd of object in the *Company* class. Object Company[2] and object Person[1] reference object Car[2] directly but they reference object Engine[3] and Body[5] indirectly. The information of an entry in the format of <sig(relationship objects in AF group), OIDs of classes in AF group, sig(relationship objects in I group), OIDs of objects in I group> will be stored in the Join Signature file as follows.

<sig(rel objs in AF group), Company[2], Person[1], Engine[3], Body[5], Sig(rel objs in I group), Car[2]>

The information of an associated entry in the Join Signature file is shown in Fig. 8.

11010110	Company[2]	Person[1]	Engine[3]	Body[5]	10001001	Car[2]

Fig. 8. An Example of an Entry in the Join Signature File

Usually, a query will be analyzed where the predicate occurs and where the target class is. The class of which the predicate occurs will be called the predicate class. Therefore, the corresponding signature will be used in according to the predicate class. If the predicate occurs for classes in AF group, the signature of relationship objects in AF group will be used, otherwise the signature of relationship objects in I group will be used instead.

An attribute's value of the predicate class in the query is transformed into a query signature S_Q. The query signature S_Q will be compared with every signature stored in entries of the corresponding signature group in the Join Signature file. Whenever the signature of object relationship matches with the query signature, it will be verified if that entry of the Join Signature file is not false drop by retrieve the information in OODB using OID of the predicate class stored in the entry. If it is a qualified object, get the OID of the target class to retrieve information from the database.

From the characteristic of the Join Signature file, it is very convenient to locate the matching objects in the predicate class. When the predicate class and the target class are specified, the associated signature group in the Join Signature file will be used for that query. Therefore, this access method is appropriate for any locations for the predicate class and target classes. Furthermore we can obtain any target objects from the remaining classes in the hierarchy that relate to the predicate object.

When compared with the forward traversal technique, all entries of the Join Signature file will be scanned and only predicate objects of the matching signature will be accessed to verify that they are not false drop. On the contrary, the forward traversal technique explores every object in the predicate class to find the predicate objects and use them to access the descendant objects. It is noticeable that the size of a Join Signature file is much less than the size of all objects in the predicate class. Furthermore, the signature can be used to filter the objects in the predicate class. Therefore, the cost of retrieval cost of the Join Signature is much less than that of the forward traversal technique.

If the reverse pointer is not applicable, the reverse traversal will have very high retrieval cost. In the reverse traversal technique, all objects in the predicate class will be accessed to locate the objects that have attribute's value in the predicate. The qualified objects are then stored in set S. Later the parent class of the predicate class is then accessed to find if there are objects that point to the objects in set S. The traversal to the ancestor class uses the similar approach. Therefore, all class between the predicate class and the target class will be accessed making high retrieval cost.

The structure of the Join Signature supports predicate on any classes in the graph aggregation hierarchy. The corresponding signature of AF group or I group will be used according to the predicate class. After getting the predicate object, the target objects from the target classes will be accessed easily. Therefore, the retrieval cost of the Join Signature is much less than that of the reverse traversal technique.

VI. Cost Model

Cost Model consists of storage and retrieval cost. The parameters listed below will be used in the analysis

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

A. The Parameters of Cost Model

Given the graph aggregation hierarchy of classes, the parameters of the cost model for implementation are listed below.

N_A: the number of ancestor classes.
N_F: the number of final classes.
N_I: the number of intermediate classes.
Q: the average number of objects in each class.
S: the size of a signature.
I: the size of an OID.
P: the average size of an object.
E: the average size of an entry in the signature file.
K: the average size of the signature file.
R: the average matching rate of a query signature.
G: Page size.

Page will be used to estimate the storage cost and the retrieval cost because it is a basic unit to access data in the secondary storage. All lengths and sizes used above are in bytes.

The following are assumptions used in the analysis.

1. All simple attributes are single-valued.
2. All complex attributes are single-valued.

The child object is always referenced by its parent object.

B. Storage Cost

The size of a signature entry E is

$$E = 2 * S + (N_A + N_F + N_I) * I. \quad (1)$$

Thus, the size of a Join Signature file is

$$K = Q * (2 * S + (N_A + N_F + N_I) * I). \quad (2)$$

The storage cost (SC) for the signature file is

$$SC = K / G. \quad (3)$$

C. Retrieval Cost

For a given query, the number of page access for the signature file is as follow.

If the predicate class is on the class of the AF group

$$RC_{AF} = \lceil (K / G) \rceil + \lceil (R_{AF} * Q * P / G) \rceil. \quad (4)$$

If the predicate class is on the class of I group

$$RC_I = \lceil (K / G) \rceil + \lceil (R_I * Q * P / G) \rceil. \quad (5)$$

where

RC_{AF} is the retrieval cost for the predicate class on AF group.

RC_I is the retrieval cost for the predicate class on I group.

$\lceil (K / G) \rceil$ are the number of pages required for scanning all entries in the signature file.

$\lceil (R_{AF} * Q * P / G) \rceil$ are the number of pages required to access the matching objects in the predicate class of AF group.

$\lceil (R_I * Q * P / G) \rceil$ are the number of pages required to access the matching objects in the predicate class of I group

Analysis

The important terms of the retrieval cost of the Join Signature file are the size of OID, the size of the signature, the number of classes in the hierarchy and the number of objects in each class. It is noticeable that the matching rate R_{AF} *and* R_I will filter the number of objects retrieved from the predicate class according to the predicate class on AF group and I group respectively. For the forward traversal technique, all objects of the predicate class must be retrieved to find the qualified objects and then follow the links to the target objects in the target class. Therefore, the retrieval cost of the Join Signature file is lower because of the smaller size of the signature and OID. Moreover, the mechanism of filter from the signature will access only the matching objects in the predicate class.

The retrieval cost of the reverse traversal is very high if there is no the reverse pointer from the predicate class to the target class. Therefore, all objects of all classes from the target class to the predicate class must be accessed.

It is not easy to use the forward traversal technique if the predicate class is the descendant class and the target class is the ancestor class. All object of the parent class of the predicate class must be accessed to examine if their nested attributes are qualified. Likewise, reverse traversal technique takes time if the predicate class is the ancestor class. However, the Join Signature can be used whether the location of the predicate class is any classes.

From the structure of the Join Signature file, all objects of classes in graph aggregation hierarchy that share the same predicate object are stored together. Therefore, it is convenient to access the associated target objects for the predicate object.

VII. Conclusion and Perspective

In this paper, joining of classes is classified into two groups, i.e. group of classes from ancestor class and final class (AF group) and group of classes from intermediate class (I group).The structure of corresponding joining objects and their signature are described. The Signature is created for object relationship to store abstract information of the joining objects.

This research emphasizes that the aggregation relationship of classes is formed as a graph. The retrieval of the Join Signature is discussed and compared with that of the forward traversal and reverse traversal technique. The formulation of cost model has constraints that all attributes have single value. Additionally, all child objects are always referenced. Finally, the storage overhead and the retrieval cost are formulated.

By using the signature technique, the filter mechanism will discard unnecessary objects so that only the possible objects will be accessed making lower retrieval cost.

The constraints given in this research should be resolved in the next research. The multi value attributes and reference

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

sharing for graph aggregation hierarchy should be considered to cope with complex access method.

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, 1989.

[2] E. Bertino and P. Foscoli, "Index Organization for Object-Oriented Database System," *IEEE Trans. on Knowledge and Data Eng.*, volume(7), pp.193-209, 1995.

[3] F. Fotouhi, T. G. Lee and W. I. Grosky," The Generalized Index Model for Object-Oriented Database Systems," *Proc.10th Phoenix Conf. on Computer and Communication*, pp.302-308, 1991.

[4] S. Choenni, E. Bertino, H. M. Blahken and T. Chang, " On the Selection of Optimal Index Configuration in OO Databases," *Proc. 10th Int'l Conf. on Data Eng.*, pp.526-537, 1994.

[5] W. C. Lee and D. L. Lee, " Short Cuts for Traversals in Object-Oriented Database Systems," *Proc. Int'l Computer Symposium*, pp. 1172-1177, 1994.

[6] W. C. Lee and D. L. Lee "Signature File Methods for Indexing Object-Oriented Database Systems," *Proc. Int'l Computer Science Conference*, pp. 616-622,1992.

[7] P. Mahatthanapiwat. " Join Signature," *Proc. The 7th National Computer Science and Engineering Conference*, pp. 169-173, 2003.

[8] P. Mahatthanapiwat "Join Signature with Reference Sharing", *Proc. Int'l Multiconference of Engineers and Computer Scientists*, pp. 619-623, 2007.

[9] P. Mahatthanapiwat " Join Signature for Tree Aggregation Hierarchy", *Proc. Int'l conference on Software, Knowledge, Information Management and Application*, pp. 120-124, 2008.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)