CARPATHIAN J. MATH.
20 (2004), No. 1, 21 - 30

Advanced Object Database Design Techniques

ADRIAN SERGIU DARABANT and ALINA CÂMPAN

ABSTRACT. Class fragmentation is an important task in the design of Distributed Object Oriented Databases (DOOD). However, fragmentation in DOOD is still at its beginnings and mostly adapted from the relational approaches. In this paper we propose an alternative approach for horizontal fragmentation of DOOD. Our method uses two different AI clustering techniques for partitioning class instances into fragments: the *agglomerative hierarchical* method and the *k-means centroid based* method. Class objects are modelled in a vector space; similarity between objects is computed using different measures. Finally, we provide quality and performance evaluations using a partition evaluator function .

1. INTRODUCTION

Advanced design of distributed Object Oriented Databases (OODB) involves *entity fragmentation* and *fragment allocation* to the sites of a distributed system. Recently these issues have been considered in [10, 6, 7], either for the complex object oriented data model, or just for flat data models.

For fragmenting a class it is possible to use two basic techniques: *vertical fragmentation* and *horizontal fragmentation*. In an Object Oriented (OO) environment, horizontal fragmentation distributes class instances into fragments. Each object has the same structure and a different state or content. Thus, a horizontal fragment of a class contains a subset of the whole class extension. Horizontal fragmentation is usually subdivided in *primary* and *derived fragmentation*.

Different approaches have been identified in solving issues regarding fragmentation, which, to a large extent, aim to extend and develop the relational fragmentation and allocation techniques to OODBs [8]. However, object data models are inherently more complex than the relational model. Features like encapsulation, inheritance, aggregation and association relations complicate the definition of the horizontal class fragmentation. There are research papers in the OO area, which claim that starting from relational fragmentation techniques brings a handicap, difficult to cover.

Algorithms for horizontal class fragmentation are proposed in [6, 3, 11]. Vertical fragmentation is addressed in [7]. Mixed fragmentation is considered in [2].

Contributions: We focus in this paper on horizontal object oriented fragmentation by using alternative methods to cluster objects into fragments. We propose new techniques for horizontal fragmentation in object-oriented databases with simple attributes and methods. They rely on AI non-supervised clustering techniques for partitioning classes into sets of similar instance objects, rather than

Received: 26.09.2004; In revised form: 12.01.2005
2000 *Mathematics Subject Classification.* 68P15, 68M14, 62H30, 68M20, 68U35.
Key words and phrases. *distributed object oriented database systems, horizontal fragmentation, clustering algorithms, fragmentation quality evaluation.*

following the traditional minimal predicate set method. We consider two clustering methods: the *agglomerative hierarchical* method and the *k-means centroid based* method [9]. Although these are well known clustering techniques, they have not been used yet in object-database fragmentation, to our knowledge.

The algorithms group objects together by their similarity with respect to a set of user queries with conditions imposed on data. Similarity (dissimilarity) between objects is defined in a vector space model and is computed using different metrics. As a result, we cluster objects that are highly used together by queries.In order to improve fragmentation quality, in the k-means algorithm we propose several methods for choosing initial cluster centroids, according to queries semantic.

The paper is organized as follows. The next section of this work presents the object data model and the constructs used in defining the object database and expressing queries. It also introduces the vector space model we use to compare objects, methods for constructing the object characteristic vectors and similarity metrics over this vector space. Section 3 presents our fragmentation algorithms. In section 4 we present a complete fragmentation example over a class hierarchy and we evaluate the quality of our fragmentation schemes by using an evaluator function.

2. Data Model

We use an object-oriented model with the basic features described in the literature [4, 1]. OO databases represent data entities as objects supporting features like inheritance, encapsulation, polymorphism, etc. Objects with common attributes and methods are grouped into classes. A class is an ordered tuple $C = (K, A, M, I)$, where A is the set of object attributes, M is the set of methods, K is the class identifier and I is the set of instances of class C. We deal in this paper only with simple attributes and methods. Simple attributes have primitive data types as their domain. Simple methods access only attributes of their class. Every object in the database is uniquely identified by an OID. Each class can be seen as a class object. Class objects are grouped in meta-classes [4].

Classes are organized in an inheritance hierarchy, in which a subclass is a specialization of its superclass. Although we deal here for simplicity only with simple inheritance i.e. a class can have at most one superclass, moving to multiple inheritance would not affect the fragmentation algorithms in any way, as long as the inheritance conflicts are dealt with into the data model. Association between an object and a class is materialized by the instantiation operation. An object O *is an instance of a class* C if C is the most specialized class associated with O in the inheritance hierarchy. An object O *is member of a class* C if O is instance of C or of one of subclasses of C. An OODB is a set of classes from an inheritance hierarchy, with all their instances. There is a special class Root that is the ancestor of all classes in the database. Thus, in our model, the inheritance graph is a tree.

An *entry point* into a database is a metaclass instance bound to a known variable in the system. An entry point allows navigation from it to all classes and class instances of its subtree (including itself). For our model, we suppose that each class is the data-base is an entry point, which means that its members are known.

In general, in our model, a *query* is a tuple with the following structure: $q = (TargetClass, Qualificationclause)$, where *Target class* - specifies the root of the class hierarchy over which the query returns its object instances and it is an entry point; *Qualification clause* - is a logical expression over the class attributes in conjunctive normal form. The logical expression is constructed using simple predicates: $attribute\theta value$ where $\theta \in \{<, >, \leq, \geq, =, \neq\}$.

Let $Q = \{q_1, \ldots, q_t\}$ be the set of all queries in respect to which we want to perform the fragmentation. Let $Pred = \{p_1, \ldots, p_q\}$ be the set of all simple predicates Q is defined on. Let $Pred(C) = \{p \in Pred | p$ imposes a condition to an attribute of class $C\}$.

Given two classes C and C', where C' is subclass of C, $Pred(C') \subset Pred(C)$. Thus the set of predicates for class C' comprises all the predicates directly imposed on at-tributes of C' and the predicates defined on attributes of its parent class C and inherited from it. We model class predicates this way in order to capture on subclasses the semantic of queries defined on superclasses. For example, given the hierarchy in FIGURE 4 , a condition "student.grade>5" imposed on Student should normally be reflected on all instances of Grad students as well (graduates are also students).

We construct the object-condition matrix for class $C, OCM(C) = \{a_{ij}, 1 \leq i \leq |Inst(C)|, 1 \leq j \leq |Pred(C)|\}$, where $Inst(C) = \{O_1, \ldots, O_m\}$ is the set of all instances of class C, $Pred(C) = \{p_1, \ldots, p_n\}$. Each line i in $OCM(C)$ is the object-condition vector of O_i, where $O_i \in Inst(C)$. We obtain from $OCM(C)$ the characteristic vectors for all instances of C. The characteristic vector for object O_i is $w_i = (w_{i1}, w_{i2}, \ldots, w_{in})$, where

$$a_{ij} = \begin{cases} 0, if\ p_j(O_i) = false \\ 1, if\ p_j(O_i) = true \end{cases}, \tag{2.1}$$

$$w_{ij} = \frac{\sum\limits_{l=\overline{1..m}, a_{lj}=a_{ij}} [(a_{lj}|a_{lj}=1)+(1-a_{lj}|a_{lj}=0)]}{|m|}$$

Each w_{ij} is the ratio between the number of objects in C respecting the predicate $p_j \in Pred(C)$ in the same way as O_i and the number of objects in C. We denote the characteristic vector matrix as $CVM(C)$. Over the set of characteristic vectors associated to all C's instances we define two similarity measures between two objects O_i and O_j:

$$sim_{cos}(O_i, O_j) = cos(w_i, w_j) = \frac{\sum\limits_{k=1}^{n} w_{ik} \times w_{jk}}{\sqrt{\sum\limits_{k=1}^{n} (w_{ik})^2} \times \sqrt{\sum\limits_{k=1}^{n} (w_{jk})^2}} \tag{2.2}$$

$$sim_M(O_i, O_j) = 1 - \frac{d_M(w_i, w_j)}{m}, \quad d_M(w_i, w_j) = \sum_{k=1}^{n} |w_{ik} - w_{jk}| \tag{2.3}$$

d_M is the Manhattan distance as defined in [9]. The second equation uses the vectorial cosine as similarity measure. We should note that all characteristic vectors have positive coordinates by definition.

3. The partitioning algorithms

In the following paragraphs we present the two fragmentation algorithms: *hierarchical agglomerative fragmentation* and *k-means non-hierarchical fragmentation.*

3.1. **The Hierarchical Agglomerative Fragmentation.** The hierarchical clustering method creates a hierarchical decomposition of the given set of data objects. We use in our paper an agglomerative variant of the hierarchical clustering algorithm [9].

Algorithm `HierachicalAggFrag` **is**
Input : `Class` C, $Inst(C)$ `to be fragmented, the similarity function` $sim : Inst(C) \times Inst(C) \longrightarrow [0,1]$, $m = |Inst(C)|$, $1 < k \leq m$ `desired number of fragments, OCM(C), CVM(C).`
Output : `The set of hierarchical clusters` $F = \{F_1, \ldots, F_k\}$
Begin
`For i=1 To` $Inst(C)$ `do` $F_i = \{w_i\}$;
$F = \{F_1, \ldots, F_m\}$;
`While` $|F| > k$ `do`
$(F_{u*}, F_{v*}) := argmax(F_u, F_v)[sim(F_u, F_v)]$;
$F_{new} = F_{u*} \cup F_{v*}$;
$F = F - \{F_{u*}, F_{v*}\} \cup \{F_{new}\}$;
`End While;`
End.

An input vector w_i quantifies the way object O_i satisfies predicates in $Pred(C)$ with respect to the way all other objects satisfy those predicates. When grouping objects (clusters), the algorithm gives priority to those two respecting most of the predicates in the same way. At each iteration the algorithm chooses the two most similar clusters and merges them into a single cluster ($argmax(F_u, F_v)[sim(F_u, F_v)]$) using the following intercluster similarity :

$$sim(F_u, F_v) = \frac{\sum_{a_i \in F_u} \sum_{b_j \in F_v} sim(a_i, b_j)}{|F_u| \times |F_v|} \tag{3.4}$$

At the end of the algorithm we always have k clusters representing the class fragments.

3.2. **The k-means non-hierarchical fragmentation.** The classical k-means algorithm takes the input parameter k and partitions a set of m objects into k clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. Cluster similarity is measured in regard to the *mean* value of

the objects in a cluster, and can be viewed as the cluster's *center of gravity (centroid)*. First, the k-means algorithm randomly selects k of the objects, each of which initially represent a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which is the most similar, based on the distance between the object and the cluster centroid. It then computes the new centroid for each cluster and redistributes all objects according to the new centroids. This process iterates until the criterion function converges. The criterion tries to make the resulting k clusters as compact and separate as possible.

Our version of the algorithm improves several aspects of the original algorithm with regard to the semantic of object fragmentation. First of all we choose as centroids the most representative objects in respect with fragmentation predicates, rather than choosing them arbitrarily. At each iteration, if an object should be placed in any of several clusters (same similarity with the centroid), we choose the cluster to which the object has maximum similarity with. We also choose as criterion function the degree of compactness/homogeneity H of all clusters. This value is the difference be-tween the maximum and minimum similarity of all pairs of objects in the cluster.

(3.5)
$$H(F) = max\{sim(a,b) | (a,b) \in F \times F, a \neq b\} - min\{sim(a,b) | (a,b) \in F \times F, a \neq b\}$$

Algorithm `k-MeansFrag is`
Input `as in HierarchicalAggFrag`
Output : `The set of clusters` $F = \{F_1, \ldots, F_f\}$, `where` $f \leq k$
Begin
  `Centr=`$\{c_1, \ldots, c_k\} = InitCentr(Inst(C), OCM(C), CVM(C), k)$;
  `F=`$\{F_i | F_i = \{c_i\}, c_i \in Centr, i = 1..k\}; F' = \emptyset$;
  `While F'<>F and H(F)<threshold_value do`
    `F'=F;`
    `For all objects` O_i `do`
      $F_{candidates}$`=`$\{argmax_{centr}(sim(O_i, c_l), l = 1..k)\}$; `(1)`
      $F_{u*} = \{argmax_{sim}(sim(O_i, F_r), F_r \in F_{candidates})\}$; `(2)`
      $F'_v = F'_v - \{O_j\}$ `where` $O_j \in F'_v$; $F'_{u*} = F'_{u*} \cup \{O_i\}$;
      $F' = F' - \{F'_l | F'_l = \emptyset\}$;
    `End For;`
    `For all` $F_i \in F$ `recalculate centroid` c_i; `End For;`
  `End While;`
End.
Function `InitCentr(Inst(C), OCM(C), CVM(C), k) is`
Begin
    `Centr=`$\emptyset$; `n=`$|Pred(C)|$;
    `For i=1 to k do`
      $c_i = argmin[d_M(a_j, u_i)], O_j \notin Centr, i \leq n$; `(3)`
      $c_i = argmin[sim(O_j, Centr)], O_j \notin Centr, i > n$; `(4)`
      $Centr = Centr \cup \{c_i\}$;
    `End For;`
    `Return Centr;`
End Function;

Function InitCentr chooses the initial centroids as described above. In line (3) u_i is identity vector of degree i, which has 1 only on i^th position and 0 on the other positions. Each u_i represents the corresponding predicate from $Pred(C)$. Line (3) chooses as centroid the closest object to u_i, i.e. the most representative object for that predicate. We should note that we can choose this way as many centroids as the number of predicates in $Pred(C)$. If we need more clusters than $|Pred(C)|$, we choose their initial centroids the objects most dissimilar to the already chosen centroids (line (4)). We try this way to minimize the impact of "losing" clusters in the following iterations. This occurs when all objects in a cluster relocate to other clusters because the initial centroid is not semantically representative to our set of predicates. We use in lines (1) and (2) the similarity of an object O_i with a cluster F_c:

$$sim(O_i, F_c) = \frac{\sum_{a \in F_c} sim(O_i, a)}{|F_c|} \tag{3.6}$$

4. Results and Evaluation

In this section we illustrate the experimental results obtained by applying our fragmentation schemes on a test object database. Given a set of queries, we first obtain the horizontal fragments for the classes in the database; than we evaluate the quality and performance of the fragmentation results. For evaluation we use a variant of the Partition Evaluator proposed by Chakravarthy in [5] for vertical relational fragmentation. The sample object database represents a reduced university database. The inheritance hierarchy is given in FIGURE 4 . The queries running on the classes of the database are given bellow.

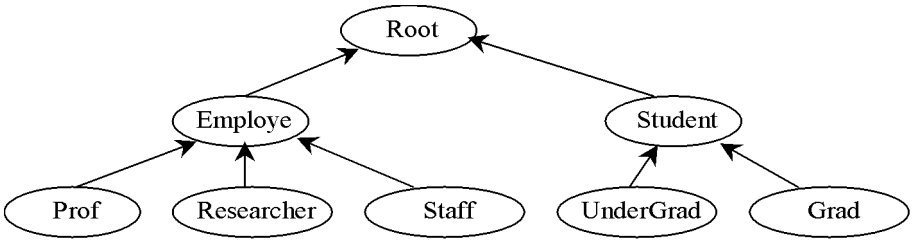

Figure 1. The database class hierarchy

q_1: This application retrieves all lecturers and teaching assistants.
q_1 =(Prof, Prof, Prof.position in ("lecturer", "teaching assistant"))
q_2: This application retrieves all professors and assistant professors.
q_2 =(Prof, Prof, Prof.position="prof." or Prof.position="assist. prof.")
q_3: This application retrieves all researchers older than 30 years.
q_3 =(Researcher, Researcher, Researcher.age≥30)
q_4: This application retrieves all researchers having published at least two papers.
q_4 =(Researcher, Researcher, Researcher.count(Reasercher.doc)≥2)

q_5: This application retrieves all graduates with grades less than 4 enrolled at the Computer Science departments.
q_5 =(Grad, Grad, Grad.grade≤4 and Grad.dept like "CS*")
q_6: This application retrieves all graduates older than 30.
q_6 =(Grad, Grad, Grad.age≥30)

The fragments obtained for *Grad* using algorithm k-meansFrag and cosine as similarity measure are: $F_1 = \{G_5\}$, $F_2 = \{G_9, G_7, G_12, G_11, G_10\}$, $F_3 = \{G_6, G_3, G_1, G_8\}$, $F_4 = \{G_4, G_13, G_2\}$. The fragments obtained for *Grad* using algorithm HierachicalAggFrag and cosine as similarity measure are: $F_1 = \{G_5\}$, $F_2 = \{G_9, G_7, G_12, G_11, G_10\}$, $F_3 = \{G_6, G_3, G_1, G_8\}$, $F_4 = \{G_4, G_13, G_2\}$. The fragments obtained for *Researchers* using the algorithm HierachicalAggFrag and Manhattan on characteristic vectors as similarity measure are: $F_1 = \{R_1, R_3, R_4\}$, $F_2 = \{R_2\}$, $F_3 = \{R_5\}$, $F_4 = \{R_6\}$. The fragments obtained for *Researchers* using algorithm k-meansFrag and cosine as similarity measure are: $F_1 = \{R_6, R_2\}$, $F_2 = \{R_1, R_3, R_5, R_4\}$.

Using the given query access frequency and other input data, the fragments above are allocated to four distributed sites. We use a simple allocation scheme that assigns fragments to sites where they are most needed. Query frequency is given bellow.

freq(q,s)	S1	S2	S3	S4	Class
q1	10	20	5	20	Prof
q2	0	10	5	25	Prof
q3	20	10	15	10	Researcher
q4	15	10	25	20	Researcher
q5	25	20	0	20	Grad
q6	30	25	20	10	Grad

First we qualitatively compare the cosine k-means fragmentation with a fully replicated database and a centralized database allocated on one of the sites.

The Partition Evaluator proposed in [5] is formed by two terms: the local irrelevant access cost (EM) and the remote relevant access cost (ER). For a class C, the EM term computes the number of non-accessed local objects in all fragments; the ER term computes the number of remote objects accessed by all queries running at each site.

$$PE(C) = EM^2 + ER^2 \quad with \tag{4.7}$$

$$EM^2(C) = \sum_{i=1}^{M} \sum_{t=1}^{T} freq_{ts}^2 * |Acc_{it}| * \left(1 - \frac{|Acc_{it}|}{|F_i|}\right) \tag{4.8}$$

$$ER^2(C) = \sum_{t=1}^{T} \min \left\{ \sum_{s=1}^{S} \sum_{i=1}^{M} freq_{ts}^2 * |Acc_{it}| * \frac{|Acc_{it}|}{|F_i|} \right\} \tag{4.9}$$

where s in EM is the site where F_i is located, while s in ER is any site not containing F_i, M is the number of clusters for class C, T is the number of queries and S is the number of sites. Acc_{it} represents the set of objects accessed by the query q_t from the fragment F_i. The smaller PE is, better fragmentation quality we have.

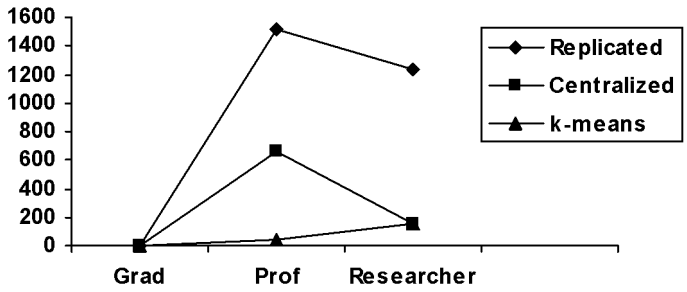

FIGURE 2. Comparative PE for k-means, full replication and centralized case.

The test results show that, generally, k-means methods perform better than hierarchical methods. This is due to the fact that with hierarchical methods, once a step is done it can never be undone. This rigidity is useful in that it leads to smaller computation costs by not worrying about combinatorial number of different choices. However, a major problem of such techniques is that they cannot correct erroneous decisions. When it comes to similarity measures, both cosine and Manhattan distinguish objects that do not respect predicates in the same way, but the differentiation refinement has different granularity for each method.

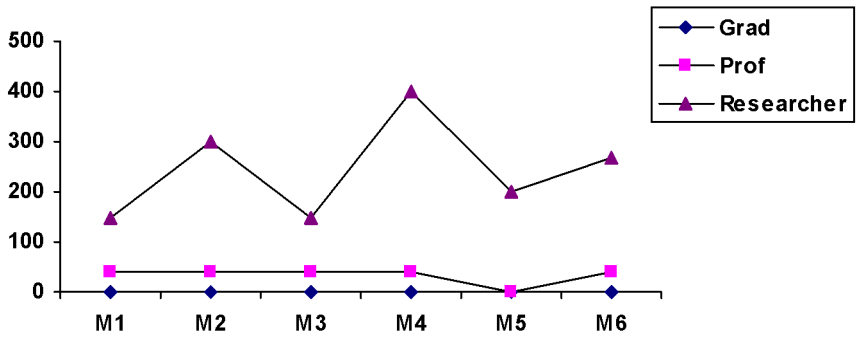

FIGURE 3. Comparison quality measures for each of our fragmentation methods.

M1	k-means cosine
M2	k-means Manhattan on object-conditions
M3	k-means Manhattan on characteristic vectors
M4	Hierarchical cosine
M5	Hierarchical Manhattan on object-conditions
M6	Hierarchical Manhattan on characteristic vectors

TABLE 1. Fragmentation methods legend

As a consequence, resulting fragments are not always similar for the same input data. Also, the experiments show that no measure behaves optimally in all cases. For example, there are particular data distributions, with perfectly separable clusters, where cosine measure is not capable of distinguishing any clusters. We have identified these particular cases and we are investigating solutions for handling them.

5. Conclusions and Future Work

In this work we prove that AI clustering methods can be effectively used in object-oriented fragmentation and we aim to extend the proposed approach to class models with complex aggregation (association) hierarchies and complex methods. Currently, we are investigating new similarity measures with improved discriminative power. We are also working on alternative evaluation techniques for fragmentation quality. We also think that we can use our clustering methods to help solving dynamic fragmentation - by capturing the semantic of potential future query changes into the initial fragmentation, so that the fragments can be adapted to those changes with smaller costs.

References

[1] Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D., Zdonik, S., *The Object Oriented Database Manifesto*, Proc. 1st Intl. Conf. on Deductive and Object-Oriented Databases, 1989

[2] Baiao, F., Mattoso, M., *A Mixed Fragmentation Algorithm for Distributed Object Ori-ented Databases*, Proc. 9th Intl. Conf. on Computing Information, Canada, 1998, 141-148

[3] Bellatreche, L., Karlapalem, K., Simonet, A., Horizontal Class Partitioning in Object-Oriented Databases, in *Lecture Notes in Computer Science*, **1308** (1997), 58-67

[4] Bertino, E., Martino, L., *Object-Oriented Database Systems; Concepts and Architectures*, Addison-Wesley, 1993

[5] Chakravarthy, S., Muthuraj, J., Varadarajan, R., Navathe S.B, An Objective Function for Vertically Partitioning Relations in Distributed Databases and its Analysis, in *Distributed and Parallel Databases*, **2**(**1**), 1993, 183-207

[6] Ezeife, C. I., Barker, K., A Comprehensive Approach to Horizontal Class Fragmentation in a Distributed Object Based System, in *Int. Journal of Distributed and Parallel Databases*, **3**(**3**), 1995, 247-272

[7] Ezeife, C., Barker, K., Vertical Class Fragmentation in a Distributed Object Based System, Technical Report 94-03, University of Manitoba, Canada, 1994

[8] Ezeife, C. I., Barker, K., *Horizontal Class Fragmentation for Advanced-Object Modes in a Distributed Object-Based System*, Proc. 9th Intl. Symposium on Computer and Information Sciences, Turkey, 1994, 25-32

[9] Han, J., Kamber, M., *Data Mining: Concepts and Techniques*, The Morgan Kaufmann Series in Data Management Systems, 2000

[10] Karlapalem, K., Navathe, S.B., Morsi, M.M.A., Issues in distribution design of object-oriented databases, in *Distributed Object Management*, Morgan Kaufmann Publishers, 1994, 148-164

[11] Savonnet, M. et. al., *Using Structural Schema Information as Heuristics for Horizontal Fragmentation of Object Classes in Distributed OODB*, Proc. IX Intl. Conf. on Parallel and Distributed Computing Systems, France, 1996, 732-737

Babes Bolyai University
Department of Computer Science
M. Kogalniceanu 1, 3400, Cluj-Napoca, Romania
E-mail address: {dadi,alina}@cs.ubbcluj.ro