AI Clustering Techniques: a New Approach to Object Oriented Database Fragmentation

Adrian Sergiu Darabant
Faculty of Mathematics and Computer Science
Babes Bolyai University – Cluj Napoca
1 M Kogalniceanu, Cluj Napoca, 3400
Romania
dadi@cs.ubbcluj.ro

Alina Campan
Faculty of Mathematics and Computer Science
Babes Bolyai University – Cluj Napoca
1 M Kogalniceanu, Cluj Napoca, 3400
Romania
alina@cs.ubbcluj.ro

Abstract – **Optimal application performance on a Distributed Object Based System requires class fragmentation and the development of allocation schemes to place fragments at distributed sites so data transfer is minimal. In this paper we present a horizontal fragmentation approach that uses the k-means centroid based clustering method for partitioning object instances into fragments. Our new method takes full advantage of existing data, where statistics are already present. We model fragmentation input data in a vector space and give different object similarity measures together with their geometrical interpretations. We provide quality and performance evaluations using a partition evaluator function.**

I. INTRODUCTION

The distribution design of an Object Oriented Database (OODB) should handle data partitioning into a cohesive set of fragments, their assignment to local processing sites and the evaluation and fine-tuning for system performance. Minimizing data transfer has been already considered by almost all distribution techniques [1], either supporting complex characteristics of the object oriented model, or just flat data models [2][6]. For fragmenting a class it is possible to use two basic techniques: horizontal fragmentation and vertical fragmentation.

We focus in this paper on horizontal object oriented fragmentation by using alternative methods to cluster objects into fragments. The object oriented data models are inherently more complex than the relational model, and complicate the definition of the horizontal class fragmentation. Different approaches have been identified in solving fragmentation issues, which, to a large extent, aim to extend and develop the relational fragmentation to OODBs. Some research papers, however, state that starting from the relational approach brings a handicap.

Related Work

Karlapalem identifies several issues and criteria for fragmenting distributed OODB horizontally and vertically [1][4][5]. Bellatreche et al. [9], Ezeife and Barker [2][7], Savonnet et. al. [10] propose algorithms for horizontal fragmentation of object classes. Ezeife uses minterm predicates identified in queries and object affinity measures with respect to these predicates to achieve primary and derived horizontal fragmentation. Savonnet et. al. present an OO distribution design methodology based on class dependency graph.

Contributions

We propose a new technique for horizontal fragmentation in object-oriented databases with simple attributes and methods. It relies on an AI non-supervised clustering method (*agglomerative hierarchical* method [3]) for partitioning classes into sets of similar instance objects, rather than following the traditional minimal predicate-set method. Although this is a well-known clustering technique, it has not been used yet in object-database fragmentation, to our knowledge.

The algorithm groups 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.

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 algorithm. In section 4 we present a complete fragmentation example over a class hierarchy and we evaluate the quality of our fragmentation schemes by using a variant of the Partition Evaluator [12].

II. DATA MODEL

We use an object-oriented model with the basic features described in the literature [8][11]. Object-oriented 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 simple methods. Every object in the database is uniquely identified by an OID. Each class can be seen in turn as a special class object. Class objects are grouped together in metaclasses.

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, moving to multiple inheritance would not affect the fragmentation algorithm in any way, as long as the inheritance conflicts are dealt with into the data model. 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.

Basic Concepts

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 sub-tree (including itself). There are usually more entry points in an OODB. In our case all classes are entry points.

In general, a *query* is a tuple with the following structure q=(Target class, Range source, Qualification clause), where:

- *Target class* -- (query operand) specifies the root of the class hierarchy over which the query returns its object instances.
- *Range source* -- a path expression specifying the source hierarchy.
- *Qualification clause* -- logical expression over the class attributes, in conjunctive normal form. The logical expression is constructed using simple predicates: *attribute* θ *value* where $\theta \in \{<,>,\leq,\geq,=,\neq\}$.

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

Given two classes C and C', where C' is subclass of C, $Pred(C')\supseteq Pred(C)$. Thus the set of predicates for class C' comprises all the predicates directly imposed on attributes 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 Fig. 3, 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, $Pred(C)=\{p_1, \ldots, p_n\}$:

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

$$w_{ij}=\frac{\sum_{l=1..m, a_{lj}=a_{ij}} a_{lj}}{|Inst(C)|} \tag{1}$$

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 $w_i=(w_{i1}, w_{i2}, ..., w_{in})$, where 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 several *pseudo-metrics*:

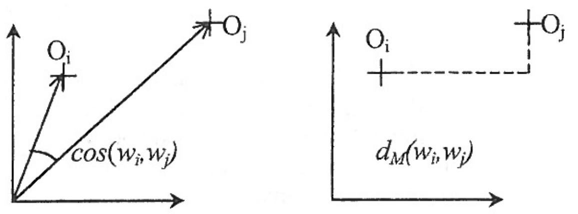

Fig. 1. Geometrical interpretation for the cosine and Manhattan similarities

$$\cos(w_i, w_j)=\frac{\sum_{k=1}^{n} w_{ik}\times w_{jk}}{\sqrt{\sum_{k=1}^{n}(w_{ik})^2}\times\sqrt{\sum_{k=1}^{n}(w_{jk})^2}} \tag{2}$$

$$d_M(w_i, w_j)=\sum_{k=1}^{n}|w_{ik}-w_{jk}| \tag{3}$$

d_M is the Manhattan distance as defined in [3]. This distance can be calculated on characteristic vectors, as well as on object-condition vectors. Cosine distance can only be applied to characteristic vectors. Given two objects O_i and O_j, we define two similarity measures between them in (4):

$$sim_{\cos}(O_i, O_j)=\cos(w_i, w_j)$$
$$sim_M(O_i, O_j)=1-\frac{d_M(w_i, w_j)}{|Inst(C)|} \tag{4}$$

According to the cosine similarity, two objects are more similar as the angle between their characteristic vectors is smaller, i.e. tends to zero. We should note that all characteristic vectors have positive coordinates by definition.

III. HIERARCHICAL AGGLOMERATIVE FRAGMENTATION

The hierarchical method creates a hierarchical decomposition of the given set of data objects. The agglomerative approach starts with each object forming a separate group. It successively merges the objects or groups closed to one another, until all of the groups are merged into one, or until a termination condition holds.

```
Algorithm HierachicalAggFrag is
Input: Class C, Inst(C) to be fragmented, the similarity function
sim:Inst(C)xInst(C)->[0,1],
m=|Inst(C)|, 1<k≤m desired number of fragments, OCM(C), CVM(C).
Output: The set of hierarchical clusters F={F_1,...,F_k}
Begin
   For i=1 To Inst(C) do Fi={w_i};
   F={F_1,...,F_m};
```

```
    While |F|>k do
        (F_u*, F_v*) :=argmax(F_u, F_v) [sim(F_u, F_v)];
        F_new=F_u*∪F_v*;
        F=F-{F_u*, F_v*}∪{F_new};
    End While;
End.
```

Fig. 2. Algorithm HierachicalAggFrag

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) together, priority will be given 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 ($argmax(F_u,F_v)[sim(F_u,F_v)]$). As similarity between two clusters F_u and F_v, we consider:

$$sim(F_u, F_v) = \frac{\sum_{a_i \in F_u} \sum_{b_j \in F_v} sim(a_i, b_j)}{|F_u| \times |F_v|} \quad (5)$$

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

IV. 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; afterwards we evaluate the quality and performance of the fragmentation results. For evaluation we use a variant of the Partition Evaluator [12].

The sample object database represents a reduced university database. The inheritance hierarchy is given in Fig. 3. The queries running on the classes of the database are given bellow:

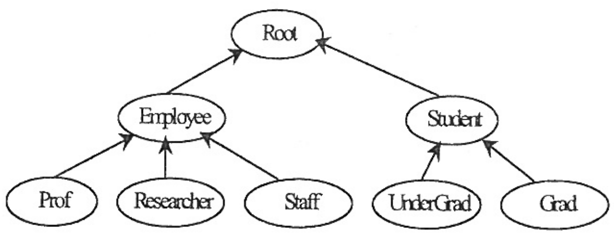

Fig. 3. The database class hierarchy

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="professor" or Prof.position = "assistant proffesor")

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.
q_5 = (Grad, Grad, Grad.grade≤4)

q_6: This application retrieves all graduates older than 30.
q_6 = (Grad, Grad, Grad.age≥30)

We only give here the *Grad* and *Researcher* instances for space and simplicity reasons.

Grad = {{a.Dept, a.Name, a.SSN, a.Born, a.Grade}, {m.age},
G_1 {CSR, Bercea Mihai, 1801229203220, 29/12/1980, 8.76}
G_2 {CSR, Bleza Ovidiu, 2850912244171, 03/09/1971, 3.00}
G_3 {M, Caciula Anamaria, 2790429080061, 29/04/1979, 9.07}
G_4 {SD, Catana Florin, 2850912244353, 01/01/1970, 3.00}
G_5 {PC, Cerba Dan, 2850912244296, 24/02/1973, 7.00}
G_6 {MI, Cigher Simona, 2800807125829, 07/08/1980, 9.06}
G_7 {MI, Cindrea Ioana, 2800924060021, 24/09/1980, 3.00}
G_8 {CSR, Cioara Danut, 1760829054671, 29/08/1976, 8.60}
G_9 {MI, Cosma Maria, 2810320060017, 20/03/1981, 3.00}
G_{10} {CSR, Damian Mircea, 1750616323929, 22/09/1976, 3.00}
G_{11} {CSM, Dani Iosif, 1761203120669, 03/12/1976, 3.00}
G_{12} {CSM, Darvas Laszlo, 1810413055099, 13/04/1981, 3.00}
G_{13} {CSR, Duhanes Dan, 2850912244193, 01/01/1970, 3.00} }
Researcher = {{a.OrgUnit, a.Name, a.SSN, a.Born, a.Doc}, {m.age},
R_1 {Algebra, Morar Oana, 2651005123456, 05/10/1973, {T4,T18,T32}}
R_2 {InfSyst, Cobarzan Claudiu, 1470120123456, 20/01/1979, {}}
R_3 {ProgrMeth, Grebla Horia, 1720501123456, 01/05/1979, {T19,P20}}
R_4 {InfSyst, Sterca Adrian, 1511123123456, 23/11/1971, {P14,P31,P32}}
R_5 {Calculus, Tofan Daniel, 1560404123456, 04/04/1976, {T16,P32}}
R_6 {InfSyst, Zeng Ioan, 1560331123456, 31/03/1972, {T17}} }

The fragments obtained by applying the hierarchical clustering algorithm with the cosine similarity measure are:

Cos-Grad: $F_1=\{G_6, G_3, G_1, G_8,\}$ $F_2=\{G_9, G_7, G_{12}, G_{11}, G_{10}\}$, $F_3=\{G_4, G_{13}, G_2\}$, $F_4=\{G_5\}$.

Cos-Prof: $F_1=\{P_2, P_{12}, P_8, P_4, P_9, P_{13}, P_{11}, P_{10}\}$ $F_2=\{P_3\}$ $F_3=\{P_{20}, P_{18}, P_6, P_7, P_1, P_{19}, P_{17}, P_{14}, P_{16}, P_{15}, P_5\}$.

Cos-Researcher: $F_1=\{R_1, R_3\}$ $F_2=\{R_6, R_2\}$ $F_3=\{R_4\}$ $F_4=\{R_5\}$.

The fragments obtained by applying the hierarchical clustering algorithm with the Manhattan similarity measure on object-conditions are:

MOB-Grad: $F_1=\{G_6, G_3, G_1, G_8\}$ $F_2=\{G_9, G_7, G_{12}, G_{11}, G_{10}\}$ $F_3=\{G_4, G_{13}, G_2\}$ $F_4=\{G_5\}$.

MOB-Prof: $F_1=\{P_3, P_1, P_{19}\}$ $F_2=\{P_2, P_{12}, P_8, P_4, P_9, P_{13}, P_{11}, P_{10}\}$ $F_3=\{P_{20}, P_{18}, P_6, P_7, P_{17}, P_{14}, P_{16}, P_{15}, P_5\}$.

MOB-Researcher: $F_1=\{R_1, R_4\}$ $F_2=\{R_2\}$ $F_3=\{R_3, R_5\}$ $F_4=\{R_6\}$.

The fragments obtained by applying the hierarchical clustering algorithm with the Manhattan similarity measure on characteristic vectors are:

MVC-Grad: $F_1=\{G_6, G_3, G_1, G_8\}$ $F_2=\{G_9, G_7, G_{12}, G_{11}, G_{10}\}$ $F_3=\{G_4, G_{13}, G_2\}$ $F_4=\{G_5\}$.

MVC-Prof: $F_1=\{P_2, P_{12}, P_8, P_4, P_9, P_{13}, P_{11}, P_{10}\}$ $F_2=\{P_3\}$ $F_3=\{P_{20}, P_{18}, P_6, P_7, P_1, P_{19}, P_{17}, P_{14}, P_{16}, P_{15}, P_5\}$.

MVC-Researcher: $F_1=\{R_1, R_3, R_4\}$ $F_2=\{R_2\}$ $F_3=\{R_5\}$ $F_4=\{R_6\}$.

Generally, we can see that the obtained fragments are tailored to query conditions. For example, F_2 from **MOB-Prof** contains all professors and all assistant professors but one (P_3). F_2 is the set of objects satisfying the first condition of q_1. The only object satisfying the other condition in q_1, P_3, is placed into fragment F_1. If we want to have fragments that exactly match queries, than we need to consider an entire qualification clause as one condition (column) in *OCM*. The fragmentation results are also influenced by k, the requested number of fragments. Objects that should normally be grouped together according to conditions or queries may remain into distinct fragments. This is because the requested number of fragments may not lead to the best fragmentation scheme.

The hierarchical clustering method does not always perform optimally due to the fact that 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.

Using the given query access frequency and other input data, the fragments above are allocated to 4 distributed sites. We use a simple allocation scheme that assigns fragments to the sites where they are most needed. Query frequency at sites is presented in TABLE 1.

TABLE 1. Access Frequencies of queries at distributed sites

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

We qualitatively compare the hierarchical fragmentation variants with a fully replicated database and a centralized database allocated on one of the sites Fig. 4, Fig. 5. We use the Partition Evaluator (*PE*) [12] to compare the fragmentation results. *PE* is composed of two terms: the local irrelevant access cost (*EM*) and the remote relevant access cost (*ER*):

$$PE(C) = EM^2 + ER^2 \tag{6}$$

$$EM^2(C) = \sum_{i=1}^{M}\sum_{t=1}^{T} freq_{ts}^2 * |Acc_{it}| * \left(1 - \frac{|Acc_{it}|}{|F_i|}\right) \tag{7}$$

$$ER^2(C) = \sum_{t=1}^{T} \min\left\{\sum_{s=1}^{S}\sum_{i=1}^{M} freq_{ts}^2 * |Acc_{it}| * \frac{|Acc_{it}|}{|F_i|}\right\} \tag{8}$$

In (7) *s* is the site where F_i is located, while in (8) *s* 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 aim is to minimize the mean square error for all fragments. Higher *PE* value means higher performance penalty and, thus, lower performance for the fragmentation scheme.

Experimental results show that both cosine and Manhattan similarity measures distinguish objects that do not respect predicates in the same way, but the differentiation refinement has different granularity for each measure. 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.

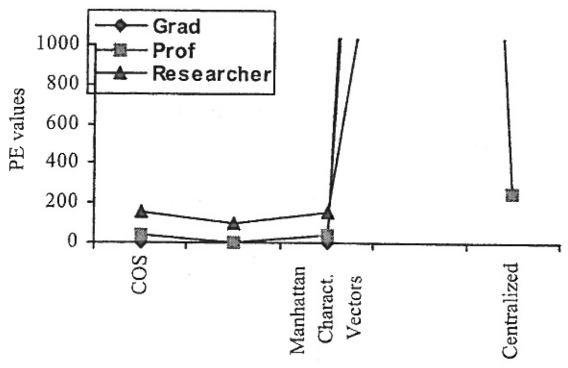

Fig. 4. Comparative quality measures for fragmentation variants, centralized and fully replicated cases

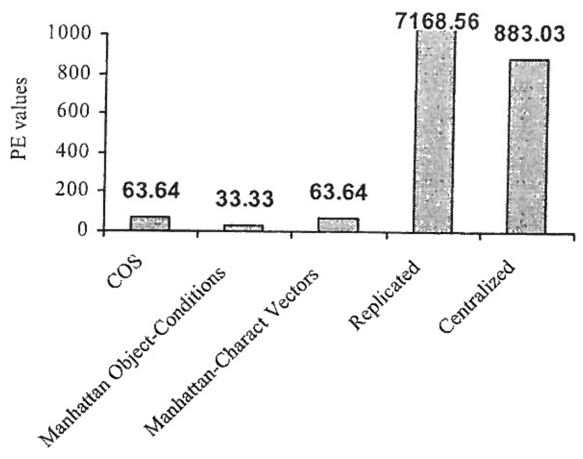

Fig. 5. Global PE values for all classes.

For example the cosine measure fails to cluster data distributions following special patterns. In these special cases the cosine measure fails to cluster even objects that are clearly separated. In TABLE 2 we show the instances of a class with 4 conditions defined on the class. Some objects only respect conditions 1,2 and 4 while the rest of them satisfy only condition 3. Thus the set of instances could be easily clustered in two classes, even by looking at objects. However the next example shows that the cosine measure is not able to do it.

TABLE 2. OCM – exceptional case

Objects/ Conditions	C_1	C_2	C_3	C_4
O_1	1	1	0	1
O_2	0	0	1	0
O_3	1	1	0	1
O_4	1	1	0	1
O_5	0	0	1	0
O_6	1	1	0	1
O_7	0	0	1	0

TABLE 3. CVM – for OCM

Objects/ Conditions	C_1	C_2	C_3	C_4
O_1	4/7	4/7	4/7	4/7
O_2	3/7	3/7	3/7	3/7
O_3	4/7	4/7	4/7	4/7
O_4	4/7	4/7	4/7	4/7
O_5	3/7	3/7	3/7	3/7
O_6	4/7	4/7	4/7	4/7
O_7	3/7	3/7	3/7	3/7

If we apply the hierarchical method with the cosine similarity and 2 clusters requested we obtain the following clusters: $F_1=\{O_1, O_2, O_3, O_4, O_5, O_6\}$, $F_2=\{O_7\}$. Instead of separating objects in two distinct clusters – the result is clearly incorrect.

For n objects following this pattern and k clusters requested, the algorithm will place the first n-k+1 objects into the first cluster and the rest will remain each in its own cluster. Objects are placed into clusters without considering the conditions they satisfy. The cosine measure between two objects in this case is zero as the two vectors have the same direction and orientation.

TABLE 4. OCM – with phantom object

Objects/ Conditions	C_1	C_2	C_3	C_4
O_1	1	1	0	1
O_2	0	0	1	0
O_3	1	1	0	1
O_4	1	1	0	1
O_5	0	0	1	0
O_6	1	1	0	1
O_7	0	0	1	0
Ph	1	1	1	1

TABLE 5. CVM – with phantom object

Objects/ Conditions	C_1	C_2	C_3	C_4
O_1	5/8	5/8	4/8	5/8
O_2	3/8	3/8	4/8	3/8
O_3	5/8	5/8	4/8	5/8
O_4	5/8	5/8	4/8	5/8
O_5	3/8	3/8	4/8	3/8
O_6	5/8	5/8	4/8	5/8
O_7	3/8	3/8	4/8	3/8
Ph	5/8	5/8	4/8	5/8

In order to solve this anomaly we use a simple stratagem. We place a phantom object in the data set to break the monotony. After fragmenting the class instances we withdraw the phantom object. The new OCM and CVM are depicted in TABLE 4 and TABLE 5. The resulting clusters are:

$F_1=\{O_1, O_3, O_4, O_6\}$, $F_2=\{O_2, O_5, O_7\}$.

The results are correct with respect to conditions defined in TABLE 2.

V. CONCLUSIONS AND FUTURE WORK

We are presenting in this paper a new AI based approach to object oriented horizontal fragmentation in the context of classes with simple attributes and simple methods. We use the hierarchical clustering algorithm with two similarity measures to fragment a set of class instances with respect to user requirements. We have identified two weak points: the incapacity of the cosine similarity measure to distinguish over data with special patterns, and the incapacity of the algorithm to correct erroneous decisions. The first problem has been solved by applying a simple technique, and we are investigating ways to reduce at minimum the effects of the other one. Nonetheless the presented method proves to be effective in practice. We aim to extend the proposed approach to class models with complex aggregation (association) hierarchies and complex methods using new similarity measures and clustering techniques.

VI. REFERENCES

[1]Karlapalem, K., Navathe, S.B., Morsi, M.M.A. – "Issues in distribution design of object-oriented databases". In M. Tamer Ozsu, U. Dayal, P. Valduriez, editors, *Distributed Object Management*, pp 148-164, Morgan Kaufmann Publishers, 1994.

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

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

[4] Karlapalem, K., Li, Q. – "Partitioning Schemes for Object-Oriented Databases", *In Proceedings of the Fifth International Workshop on Research Issues in Data Engineering-Distributed Object Management*, pp 42–49, Taiwan, 1995.

[5] Karlapalem, K., Li, Q., Vieweg, S. – "Method Induced Partitioning Schemes in Object-Oriented Databases", *In Proceedings of the 16th Int. Conf. on Distributed Computing System* (ICDCS'96), pp 377–384, Hong Kong, 1996.

[6] Ravat, S. – "La fragmentation d'un schema conceptuel oriente objet", *In Ingenierie des systemes d'information* (ISI), 4(2), pp 161–193, 1996.

[7] Ezeife, C.I., Barker, K. – "Horizontal Class Fragmentation for Advanced-Object Modes in a Distributed Object-Based System", *In the Proceedings of the 9th International Symposium on Computer and Information Sciences*, Antalya, Turkey, pp 25-32, 1994.

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

[9] Bellatreche,L., Karlapalem, K., Simonet, A. – "Horizontal Class Partitioning in Object-Oriented Databases", *In Lecture Notes in Computer Science*, volume 1308, pp 58–67, Toulouse, France, 1997.

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

[11] Baiao, F., Mattoso, M. – "A Mixed Fragmentation Algorithm for Distributed Object Oriented Databases", *In Proc. Of the 9th Int. Conf. on Computing Information*, Canada, pp 141-148, 1998.

[12] Darabant, A.S., Campan, A.– "Hierarchical AI Clustering for Horizontal Object Fragmentation", *In Proc of Int. Conf. of Computers and Communications*, pp. 117-122, May, 2004.