Hierarchical Clustering in Object Oriented Data Models with Complex Class Relationships

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

Octavian Cret
Computer Science Department
Technical University – Cluj Napoca
15 C-tin Daicoviciu, Cluj Napoca
Romania
cret@cs.utcluj.ro

***Abstract* – Class fragmentation is an essential phase in the design of Distributed Object Oriented Databases (DOODB). Horizontal and vertical fragmentation are the two commonly used fragmentation techniques. We propose here two new methods for horizontal fragmentation of objects with complex attributes. They rely on AI clustering techniques for grouping objects into fragments. Both methods take into account the inheritance and aggregation/association hierarchies. We provide quality and performance evaluations using a partition evaluator function.**

I. INTRODUCTION

The distribution design phase in 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. There are two basic fragmentation techniques: vertical and horizontal. 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.

Existing OO fragmentation approaches are usually inspired from the relational fragmentation techniques. The OO model is inherently more complex than the relational model. Inheritance, polymorphism, class aggregation and association all induce complex relations between classes in an object oriented database. In the simplest OO model classes have only attributes with scalar types (simple attributes) and all methods refer local class attributes (simple methods). Classes whose attributes have complex types/other classes as their domain (complex attributes) are the next step towards a real OO model.

Related Work

Fragmentation methods for OODB environments, or flat data models have been generally considered in Karlapalem [2], Ezeife [3], Karlapalem [4][5]. Ravat [6] uses the Bond Energy Algorithm (BEA) for vertical and horizontal fragmentation. Ezeife [7] presents a set of algorithms for horizontally fragmenting models with simple attributes/methods and complex attributes/methods. She is using the algorithm developed in Ozsu [8]. Bellatreche et al. [9] propose a method that emphasizes the role of queries in the horizontal fragmentation.

We have already discussed an alternative fragmentation method for OO models with simple attributes and simple methods in [12], based on AI clustering techniques.

Contributions

We propose a new technique for horizontal fragmentation in object-oriented databases with complex attributes. Fragmentation in complex OO hierarchies is usually performed in two steps: primary fragmentation and derived fragmentation. Primary fragmentation groups class instances according to a set of class conditions [12] imposed on their simple attributes. Derived fragmentation takes into account the class relationships (aggregation, association). It groups instances of a class in fragments according to the fragmentation of the related classes.

We propose an algorithm that unifies the two fragmentation steps into a single step. Both: class conditions and class relationships are modelled together in a vector space. Each object is represented as a vector and we use a hierarchical agglomerative clustering algorithm for separating clusters (fragments) of objects.

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 scheme 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. Every object in the database is uniquely identified by an OID. Each class can be seen in turn as a class object. Class objects are grouped together in metaclasses. This allows us to consider classes as being instances of higher-level classes that describe the database schema. This way the database schema is self-describing.

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.

An *entry point* into a database is a meta-class 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.

Given a complex hierarchy H, a *path expression* P is defined as $C_1.A_1. \ldots A_n$, n≥1 where: C_1 is an entry point in H, A_1 is an attribute of class C_1, A_i is an attribute of class C_i in H such that C_i is the domain of attribute A_{i-1} of class $C_{i-1}(1 \le i \le n)$. In the general case, A_i can be a method call. If i<n, then A_i must return a single complex type value (an object).

As presented in [12], a *query* is a tuple with the following structure: q=(Target class, Range source, Qualification clause).

- *Target class* -- (query operand) specifies the root of the class hierarchy over which the query returns its object instances.
- *Range source* -- a path expression starting from an entry point and specifying the source class hierarchy.
- *Qualification clause* -- logical expression over the class attributes and/or class methods, in conjunctive normal form. The logical expression is constructed using atomic predicates: *path_expression θ value* where θ ∈{<,>,≤,≥,=,≠, in,⊃,⊇}.

III. VECTOR SPACE MODELLING

Primary Fragmentation Modelling

We denote by $Q=\{q_1, \ldots, q_l\}$ 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 atomic predicates Q is defined on. Let $Pred(C)=\{p \in Pred \mid p$ impose a condition to an attribute of class C or to an attribute of its parent$\}$. Given the predicate $p \equiv C_1.A_1. \ldots A_n \, \theta \, value$, $p \in Pred(C_n)$, if class C_i is the complex domain of A_{i-1}, i=1..n, and A_n has a complex type or simple type.

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 [12].

We construct the object-condition matrix for class C, $OCM(C) = \{a_{ij}, 1 \le i \le |Inst(C)|, 1 \le j \le |Pred(C)|\}$, where $Inst(C) = \{O_1, \ldots O_m\}$ is the set of all instances of class C, $Pred(C) = \{p_1, \ldots, p_n\}$:

$$a_{ij} = \begin{cases} 0, if\ p_j(O_i) = false \\ 1, if\ p_j(O_i) = true \end{cases} \qquad w_{ij} = \frac{\sum_{l=1..m, a_{lj}=a_{ij}} a_{lj}}{m} \qquad (1)$$

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 $w_i = (w_{i1}, w_{i2}, \ldots, w_{in})$, 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 $CVM(C)$ [12].

Derived Fragmentation Modelling

We have captured so far all characteristics of simple attributes and methods. We need to express the class relationships in our vector space model. We first model the aggregation and association relations.

Given two classes C_O (owner) and C_M (member), where C_M is the domain of an attribute of C_O, a path expression traversing this link navigates from instances of C_O to one or more instances of C_M. When fragmenting C_O we should take in account the fragmentation of C_M. We want to place in the same fragment of C_O objects aggregating instances from a fragment of C_M. Objects of a fragment of C_O should aggregate as much as possible objects from the same fragment of C_M.

Let $\{F_1, \ldots F_k\}$ be the fragments of C_M. We denote by $Agg(O_i, F_j) = \{O^m \mid O^m \in F_j, O_i$ *aggregates* $O^m\}$.

Given the set of fragments for C_M, we define the *attribute-link induced object-condition vectors for derived fragmentation* as $ad_i = (ad_{i1}, ad_{i2}, \ldots, ad_{ik})$, where each vector component is expressed by the following formula:

$$ad_{ij} = \operatorname{sgn}\left(\left|Agg(O_i, F_j)\right|\right) \qquad (2)$$

For an object $O_i \in Inst(C_O)$ and a fragment F_j of C_M, ad_{ij} is 1 if O_i is linked to at least one object of F_j and is 0 otherwise.

Given the set of fragments for C_M, we define the *attribute-link induced characteristic vectors for derived fragmentation* as $wd_i = (wd_{i1}, wd_{i2}, \ldots, wd_{ik})$, where each vector component is expressed by one of the following formulas:

$$wd^1_{ij} = \frac{\left|Agg(O_i, F_j)\right|}{\left|\bigcup_{O_l \in Inst(C_1)} Agg(O_l, F_j)\right|} \qquad (3)$$

$$wd^2_{ij} = \frac{\left|\left\{O_l \in Inst(C_1) \mid \operatorname{sgn}\left(\left|Agg(O_l, F_j)\right|\right) = \operatorname{sgn}\left(\left|Agg(O_i, F_j)\right|\right)\right\}\right|}{\left|Inst(C_1)\right|} \qquad (4)$$

w^1_{ij} gives the number of objects in fragment F_j of class C_M linked to O_i divided to the number of objects in fragment F_j linked to instances of C_O. Two objects O_i and O_l

aggregating the same proportion of objects in F_j will be candidates to be placed in the same fragment of C_O.

Each w^2_{ij} component gives the percentage of objects in C_O that aggregate in the same way as O_i objects from F_j. Two objects O_i and O_l are said to aggregate in the same way F_j if they are both either linked or not linked with objects from F_j. According to the second criteria, two objects are candidate to be placed in the same fragment of C_O in respect to F_j if they are both related in the same way to F_j.

The attribute-link induced characteristic vectors for all objects of a class C_O will be expressed using only one of the two criteria.

Usually, the fragmentation of a class C_O is performed in two steps: primary fragmentation, according to query conditions, and derived fragmentation, according to the fragments of the member or owner classes. We merge the two phases into one single step capturing the semantic of both primary and derived fragmentations. For this we unify the characteristic vector and the attribute-link induced characteristic vectors for each object O_i of the class C_O and we obtain the *extended characteristic vector*.

If the class C_O is linked with classes $C_{M1}, C_{M2}, ..., C_{Mp}$, the *extended characteristic vector* we_i for object $O_i \in Inst(C_O)$ is obtained by appending the attribute-link induced characteristic vectors of $C_{M1}, C_{M2}, ..., C_{Mp}$ to the characteristic vector of O_i.

The *extended object-condition vector* ae_i for an object O_i is obtained in the same way by appending its attribute-link induced object-condition vectors to its object-condition vector.

We denote by *EOCM(C)* and *ECVM(C)* the extended object-condition and characteristic matrices for class *C*.

Similarity between objects

The aim of our method is to group into a cluster those objects that are similar to one another. Similarity between objects is computed using the following *pseudo-metrics*:

$$\cos(we_i, we_j) = \frac{\sum_{k=1}^{n} we_{ik} \times we_{jk}}{\sqrt{\sum_{k=1}^{n} (we_{ik})^2} \times \sqrt{\sum_{k=1}^{n} (we_{jk})^2}} \quad (5)$$

$$d_M(we_i, we_j) = \sum_{k=1}^{n} |we_{ik} - we_{jk}| \quad (6)$$

Given two objects O_i and O_j, we define two similarity measures between them in (7):

$$sim_{\cos}(O_i, O_j) = \cos(we_i, we_j)$$
$$sim_M(O_i, O_j) = 1 - \frac{d_M(we_i, we_j)}{|Inst(C)|} \quad (7)$$

Manhattan similarity is well defined for every pair of object-condition or characteristic vectors. The cosine similarity, however, is not defined for every two object-condition vectors. For extended vectors that have all components zero the cosine similarity measure is not defined. However if we look at the semantic of the characteristic vectors we can see that all components zero means that the object is not returned by any of the application queries and does not refer other objects. It can be referred, however, by other objects.

We should note that all characteristic vectors have positive coordinates by definition.

III. HIERARCHICAL AGGLOMERATIVE FRAGMENTATION

We apply the same algorithm we have used to fragment classes with simple attributes and methods ([12]).

```
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, EOCM(C), ECVM(C).
Output: The set of hierarchical clusters F={F1,...,Fk}
Begin
   For i=1 To Inst(C) do Fi={Oi};
   F={F1,...,Fm};
   While |F|>k do
      (Fu*,Fv*):=argmax(Fu,Fv)[sim(Fu,Fv)];
      Fnew=Fu*∪Fv*;
      F=F-{Fu*,Fv*}∪{Fnew};
   End While;
End.
```

Fig. 1. Algorithm HierachicalAggFrag

The *main part* of input vector we_i quantifies the way object O_i satisfies predicates in *Pred(C)* with respect to the way all other objects satisfy those predicates. The *extended part* of the vector quantifies the way an object O_i *aggregates/relates* to objects in clusters of the member classes. 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 the average similarity of all pairs of objects:

$$sim(F_u, F_v) = \frac{\sum_{a_i \in F_u} \sum_{b_j \in F_v} sim(a_i, b_j)}{|F_u| \times |F_v|} \quad (8)$$

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;

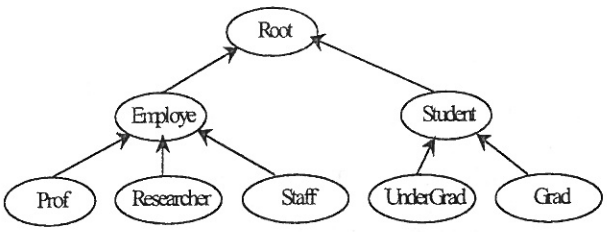

Fig. 2. The database class hierarchy

afterwards we evaluate the quality and performance of the fragmentation results. We should note that the order in which classes are fragmented is significant as it captures the semantic of query path expressions into the fragmentation process [13].

The sample object database represents a reduced university database. The inheritance hierarchy is given in Fig. 2 and the aggregation/association graph is shown in Fig. 3. The queries running on the classes of the database are given bellow:

q_1 = (Grad, Faculty.Dept.Student, Grad.Supervisor.OrgUnit.Name in ("ProgrMeth", "InfSyst"));
q_2 = (UnderGrad, Faculty.Dept.Student, UnderGrad.Dept.Name like "CS%" and UnderGrad.Grade between 7 and 10)
q_3 = (UnderGrad, Faculty.Dept.Student, (UnderGrad.Dept.Name like "Math%" or UnderGrad.Dept.Name like "CS%") and UnderGrad.Age()>=24)
q_4 = (Researcher, Doc.Person, Researcher.count(Reasercher.doc) ≥2)
q_5 = (Prof, Faculty.OrgUnit.Employee, Prof.OrgUnit.Name in ("ProgrMeth", "InfSyst") and Prof.salary>=40000)
q_6 = (Prof, Doc.Person., Prof.Paper.Publisher in ("IEEE" , "ACM") and Prof.Position="prof")
q_7 = (TechReport, Doc, TechReport.year>1999)
q_8 = (Set(Student.Dept), Person, Student.Grade<5)
q_9 = (Employee, Person, Employee.salary>35000)
q_{10} = (Grad, Person, Grad.count(Grad.Paper)>=1)
q_{11} = (Student, Person,Student.Dept.Name like "CS%")
q_{12} = (Student, Person,Student.Dept.Name like "Math%")
q_{13} = (Staff, Person, Staff.salary>12000)
q_{14} =(Person, Person, Person.Age()>30)

In Fig. 3 the links between Doc and Person should be inherited by all subclasses of Person and Doc. This is graphically represented in the figure by the dotted arrows. Similar inherited links are present for other classes in this graph (not represented here). The motivation for aggregation/association inheritance is presented in [13].

For measuring the fragmentation quality we determine

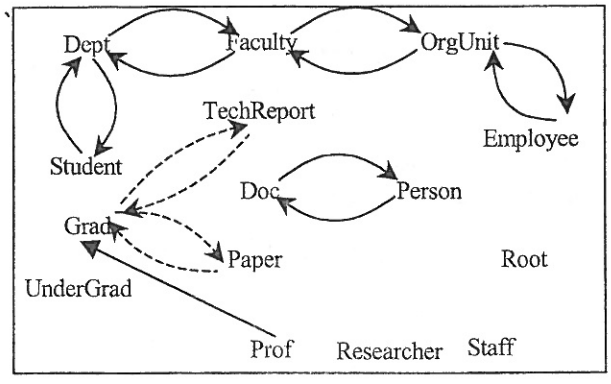

Fig. 3. The database aggregation/association graph

the cost of remote accesses and the cost of local irrelevant accesses in each fragment. In order to calculate the access cost to the fragmented database we need to allocate the fragments to the nodes of a distributed system. The cost formulas are:

$$PE(C) = EM^2 + ER^2 \quad (9)$$

$$EM^2(C) = \sum_{i=1}^{M}\sum_{t=1}^{T} freq_{ts}^2 * |Acc_{it}| * \left(1 - \frac{|Acc_{it}|}{|F_i|}\right) \quad (10)$$

$$ER^2(C) = \sum_{t=1}^{T} \min\left\{\sum_{s=1}^{S}\sum_{i=1}^{M} freq_{ts}^2 * |Acc_{it}| * \frac{|Acc_{it}|}{|F_i|}\right\} \quad (11)$$

The EM term calculates the local irrelevant access cost for all fragments of a class. ER calculates the remote relevant access cost for all fragments of a class. Acc_{it} represents the set of objects accessed by query t from fragment F_i. $freq_{ts}$ is the frequency of query t running on site s. In (10) s is the site where F_i is located, while in (11) 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 [12].

The fragmentation is better when the local irrelevant costs and the remote relevant access costs are smaller. Each term of PE calculates in fact the average square error of these factors. Globally, PE measures how well fragments fit the object sets requested by queries.

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 sites where they are most needed. Query frequency at sites is presented in TABLE 1.

We qualitatively compare the hierarchical fragmentation variants in Fig. 4. M1 conforms to eqn (3) while M2 conforms to eqn (4) for expressing derived fragmentation. In Fig. 5 we compare our methods with a fully replicated database, and a centralized database allocated on one of the sites.

TABLE 1. Access Frequencies of queries at distributed sites

Freq(q,s)	S1	S2	S3	S4
Q1	0	10	5	20
Q2	0	10	5	25
Q3	20	0	15	10
Q4	15	10	5	0
q5	25	20	0	20
q6	30	0	20	10
q7	30	25	0	10
q8	10	0	0	10
q9	20	20	10	0
q10	15	25	0	0
q11	5	10	5	0
q12	0	0	0	10
q13	15	0	0	5
q14	20	5	0	0

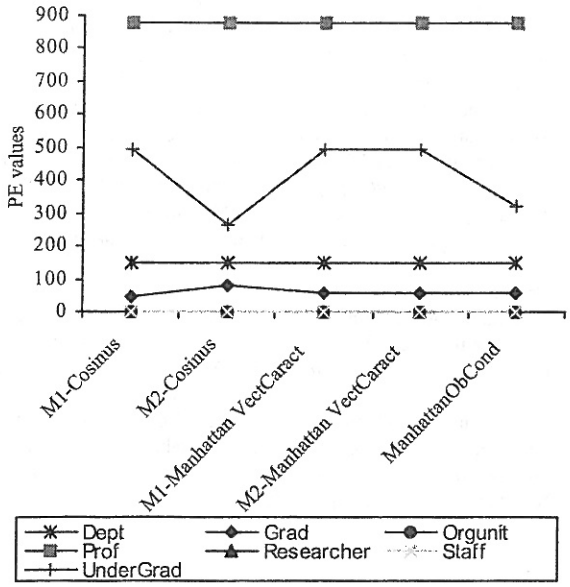

Fig. 4. Comparative quality measures for fragmentation variants

Fig. 6 compares the fragmentation performed both primary and derived, with primary only fragmentation. Labels starting with P (e.g. P-Cosine) denote the primary only fragmentation methods. It can be seen that exploiting class relationships in derived fragmentation improves the fragmentation quality.

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.

Statistically the Manhattan measure applied on object-condition vectors performs better than the other measures. In particular cases however the Manhattan measure is over-performed by the other measures. This is because in

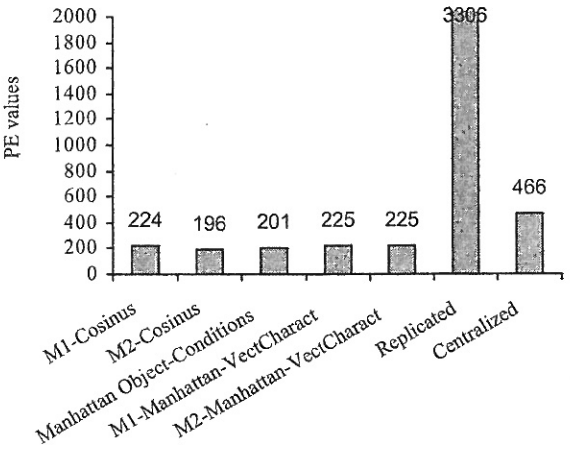

Fig. 5. Global PE values for all our methods, a fully replicated and a centralised database version.

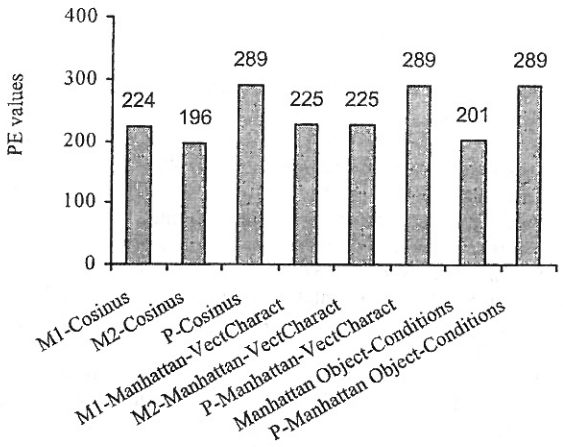

Fig. 6. PE values for complex attribute fragmentation and primary fragmentation

the hierarchical algorithms the order of handling clusters/objects has a strong influence in fusing clusters. A misplaced decision at early stages of the algorithm is carried on trough the end, accumulating more errors.

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.

V. CONCLUSIONS AND FUTURE WORK

We present in this paper an application of AI clustering methods to object oriented horizontal fragmentation of classes with complex attributes. We use the hierarchical clustering algorithm with two similarity measures to fragment a set of class instances with respect to user requirements. As with the case of classes with simple attributes we have identified a weak point: the incapacity of the algorithm to reconsider wrong cluster fusing decisions. The unpredictable character of propagating errors makes this algorithm inadequate to compare the proposed similarity metrics. Nonetheless the presented method proves to be effective in practice. We aim to apply new clustering techniques in order to avoid the undesirable effects of the hierarchical method. We also investigate the modelling of complex methods and their influence on class fragmentation.

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*, Oradea, pp 117-122, May, 2004 .

[13] Darabant, A.S, Campan, A. – "Optimal Class Fragmentation Ordering in Object Oriented Databases", *In Studia Universitatis Babes Bolyai Informatica*, Volume XLIX, Number 1 (2004), to appear, 2004.