FRAGMENTATION TECHNIQUES FOR DISTRIBUTED OBJECT-ORIENTED DATABASES

By

ELZBIETA MALINOWSKI

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

1996

Dedicated to My Unique Husband,
Children, and Parents

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Sharma Chakravarthy, for giving me the opportunity to work in such a new area - distributed object-oriented databases - and for his continual support and encouragement. I would like to express my gratitude to my supervisory committee, Dr. Stanley Su and Dr. Herman Lam, for their useful suggestions and discussions.

Also, I would like to thank my family, especially my husband and children, for the unlimited amounts of patience, support, and extraordinary encouragement that they gave me in difficult moments.

TABLE OF CONTENTS

Abstract of Thesis
Presented to the Graduate School of the University of Florida
in Partial Fulfillment of the Requirements for the
Degree of Master of Science

FRAGMENTATION TECHNIQUES FOR DISTRIBUTED OBJECT-ORIENTED DATABASES

By

Elzbieta Malinowski

August, 1996

Chairman: Dr. Sharma Chakravarthy
Major Department: Computer and Information Sciences and Engineering

Design of distributed object-oriented databases inherits the design problems from the relational distributed databases and presents additional difficulties related to schema represented by classes and their possible complicated hierarchy structure. Unlike the relational case, access to data by user-defined methods further complicates the problem.

This thesis addresses issues related to design of distributed object-oriented databases: fragmentation. Specifically, this thesis investigates vertical fragmentation of objects by extension and generalization of previous work on relational databases. Even though there are several works related to distributed object-oriented databases, few of them analyze the issues related to fragmentation.

First, we describe new matrices for capturing the semantics used for object-oriented databases. Then, we develop techniques for fragmentation of object-oriented databases and include the descriptions of algorithms for objects with: simple attributes and simple methods, complex attributes and simple methods, simple attributes and complex methods, and finally complex attributes and complex methods.

A new objective function (termed PEOO for Partition Evaluator for object-oriented databases) is derived. It is shown that this PEOO is a generalization of the PE derived for relational databases. Also, a exhaustive algorithm is used to evaluate this objective function and obtain optimal partitioning.

This PEOO can be used not only to obtain fragmentation scheme, but also to evaluate other partitioning techniques proposed in the literature.

Finally, we present several experiments to show the correctness of our approach.

At the end, we indicate how our PEOO can be adopted to a system where additional information is available (e.g. allocation information, transmission cost, and so forth).

CHAPTER 1
INTRODUCTION

The wide acceptance of the relational approach in data-processing applications and the continuous improvement of commercially existing relational databases has increased the interest for using database in non-conventional applications, such as computer-aided design (CAD), geographic information systems, image, and graphic database systems. However, these systems manage complex structures using non-traditional data types that cannot be easily represented using the relational model [32]. Consequently, the object-oriented approach has become popular, not only in the areas of programming and system design, but also in database community. The possibility of using user-defined classes and representing to the user the complex objects in the same way as simple, motivates one to intensify the research activity to develop object-oriented databases. Moreover, the wide acceptance of computer networks and advances in communication technology have changed the direction of relational database development. The distributed database (DDB) systems have become more important. They reflect more naturally the structures of many organizations and enables the possibility to efficiently share data [3].

The relational distributed databases have been investigated for many years. According to Karlapalem et al. [18] two factors, I/O operations and data transfer, are the most important for the performance of the applications in the distributed database systems. The performance of a DBMS can be improved if adequate distribution design including fragmentation, replication and allocation are applied. The relational approach distinguishes two kinds of fragmentation: horizontal and vertical. Also the hybrid fragmentation is considered another way to partition the data. There are many algorithms developed for horizontal and vertical fragmentation. Each of them has its own criteria to determine the best fragmentation scheme for the relational system being considered [5, 6, 7, 25, 28, 29].

Although much research has been dedicated to the issues of object-oriented databases, little has been related to the distribution of the objects and specifically to their fragmentation. Ezeife and Barker [10] specify benefits of fragmentation in relational databases and mention that they should be recognized in distributed object-oriented database (DOODB), too. However, there is an important aspect that makes the development of the algorithms more complicated: for a relational approach the granularity of partition was easy to identify (attributes for vertical partitioning, instances/tuples for horizontal). In contrast, in object-oriented databases, different criteria could be considered for fragmentation: hierarchical structure, "affinity" of classes, "affinity" of objects, frequency of usage of complex objects, and so forth.

1.1 Proposed Problem

Even though significant difference exist between relational and object-oriented databases, it should be clear that a relational approach can be seen (ignoring for simplicity the existence of the methods) as a special case of an object-oriented system without a class hierarchy and complex attributes (attributes that include the reference to other attributes). If this is the case, many of the algorithms developed for relational databases can be generalized and applied to object-oriented databases as one of them presented in this thesis: the problem of vertical fragmentation in distributed object-oriented databases.

The vertical fragmentation can improve transaction processing cost by decreasing the communication cost for accessing remote attributes in a distributed environment.

This thesis presents a new partition evaluator for an object-oriented database that can be applied either to obtain vertical fragmentation schemes or to evaluate “goodness” of other fragmentations. This partition evaluator is based on the one developed for relational databases and presented by Chakravarthy et al. [4].

1.2 Contributions

This thesis makes several contributions to the vertical fragmentation in DOODB:

1. A new Partition Evaluator for Distributed Object-Oriented Database (termed PEOO) is developed. It is a generalization of Partition Evaluator (PE) for vertical partitioning used in a relational approach described by Chakravarthy et al. [4]. This PEOO can be used as a tool for future fragmentation algorithms to compare the ”goodness” of the fragmentation schemes in object-oriented databases in the same way as PE can be used in relational databases.

2. New matrices that represent method and attribute usage are specified. Also, the rules for calculating the values of Attribute Usage Matrix for DOODB are presented.

3. The distinction between update and retrieve operations are provided through the method-attribute usage matrix.

4. Not only is the static information about the hierarchical structure of the object-oriented database system considered, but also the frequencies of used methods and attributes are used for establishing the fragmentation scheme. This allows the designer to distribute the attributes according to their usage and not only to static structural information.

5. The experiments are developed showing the generalization of PEOO over PE. They confirm the assumption that some techniques presented for a relational approach can be generalized and used for the object-oriented approach.

6. A hypothesis of the elimination of complex attributes from the fragmentation algorithm without affecting the optimality of solution is provided. The validity of this hypothesis is demonstrated using several examples.

7. Different levels of granularity of vertical fragmentation are proposed.

1.3 Thesis Organization

Chapters 2 and 3 discuss the issues related to the vertical partitioning in relational and object-oriented databases. Also, an overview of related work is provided in these chapters.

Chapter 4 describes the information requirements and their representation for partitioning in object-oriented database systems. Also, this chapter presents descriptions of modeling of object hierarchy for vertical partitioning.

In chapter 5 the process of development of the objective function for vertical partitioning is shown.

Several examples of small object-oriented schemes, their partitioning, minimum cost, and behavior of partition evaluator are presented in chapter 6.

Chapter 7 contains the conclusions and a list of possible extensions of this work.

CHAPTER 2
VERTICAL PARTITIONING IN RELATIONAL DATABASES

2.1 Vertical Partitioning

In a centralized database the data is stored together with all attributes that define the tuple. When this data is retrieved by some query, all attributes are loaded into main memory, even though only part of them is needed. If the number of retrieved pages is high, the rate of nonrequested attributes over the requested can also be high. If we can have in the tuple only the attributes requested by this query, the number of retrieved pages will be smaller. Therefore, if the file is partitioned in the way that the attributes which are most requested together are placed together, the number of accessed pages will be reduced. This fragmentation of attributes, called vertical fragmentation, can improve the performance of a centralized database. Moreover, the bigger effect of this partitioning can be seen in the case of distributed databases.

According to Özsu and Valduriez, "distributed databases is a collection of multiple, logically interrelated databases distributed over a computer network" ([30], p.4). It is important to observe two aspects from this definition: distributed database systems must have the possibility of communication between users (network), and data saved in different nodes of the network must be related. If the query in one node retrieves the tuples from another node, the cost of having nonrequested attributes in the tuple is higher than in the case of a centralized database because of the necessity to send these tuples over the network. This communication cost can be decreased if adequate vertical fragmentation of the data is applied.

Vertical partitioning allows the designer to group attributes of each relation into smaller records for the improvement of the efficiency of database transactions. However, the number of possible partitions of m attributes is equal to the m*th* Bell number; for a large m this number is m^m [28]. To deal with this number of solutions is almost impossible. To make this number smaller, much of the research uses the heuristic approach. One of the most used assumptions is that the partitioning has to be made considering that each fragment must be "closely matched" to the requirements of the transaction [28]. This means, in an ideal case, the fragment may contain only attributes, which are accessed by the transaction performing on this node and this transaction does not need attributes from other remote nodes. However, in distributed databases it is almost impossible to achieve this ideal case. The best partitioning of the attributes for one transaction is not necessarily the best for the other. The feasible goal is to maximize the performance of all transactions taking them as a group. Therefore, the designer must make a "trade-off," ensuring the minimum possible accesses of attributes located in remote sites for all transactions executed in this site.

2.2 Related Work

Currently, there are a considerable numbers of algorithms for vertical partitioning in relational databases.

Hoffer and Severance [14], based on the bond energy algorithm (BEA) presented by McCormick et al. [27], cluster together the attributes according to their pairwise affinity. The clusters obtained after applying BEA to the affinity attribute matrix have the characteristic that every pair of objects within clusters carries a large number of "affinity" and between the clusters this number is small. They provide the ordering of the attributes, leaving the creation of the partitions to the designer.

Hammer and Niamir [13] consider the partitioning problem as heuristic in its nature and provide grouping and regrouping of the attributes. They use the file cost estimator to check the "goodness" of obtained solution. The grouping starts assigning each of the attributes to the different fragments. In each iteration all possible grouping of these fragments is considered and the one with the maximum improvement is chosen. Regrouping is used to achieve some additional improvement moving the attributes between fragments.

Navathe et al. [28] propose the top-down approach for vertical fragmentation. They present the two-step partitioning algorithm. In the first step they use attribute usage matrix, which is recalculated to attribute affinity matrix. Then, they perform the transformation of this matrix according to BEA. Using an empirical objective function, they perform the iterative binary partitioning. In the second step, they consider the cost related to physical storage and use this cost for refinement of the solution obtained in the first step.

Cornell and Yu [7] extend the approach proposed by Navathe et al. [28] and in addition consider the access cost. They indicate that besides the transaction frequency, other aspects such as the kind of join method applied, average number of read pages, and number of used buffers are also important to consider in the algorithms for vertical fragmentation. Their algorithm uses specific physical features and allows the designer to obtain fragmentation that also decreases the number of disk accesses.

Navathe and Ra [29] describe a new algorithm for attribute clustering that uses a graph representation of an attribute affinity matrix. This graph, called affinity graph, serves as a base to form a spanning tree and to find cycles. These cycles will be considered the future fragments. They claim that this algorithm reduces complexity, allowing the designer to generate all fragments in one iteration without the necessity of using an empirical objective function.

Chu [5], and Chu and Ieong [6] use the transaction semantic instead of frequencies of attributes for vertical fragmentation. Moreover, in contrast to algorithm presented by Cornell and Yu [7], both algorithms select the best access method.

Chakravarthy et al. [4] establish function that can measure "goodness" of obtained fragments. This function can be either used for the partitioning process (exhaustive search) or applied after using some other partitioning algorithms. They argue that the AUM instead of AAM should be used in the process of fragmentation, because in AAM only the "affinity" between pairs of attributes can be reflected.

CHAPTER 3
OBJECT-ORIENTED APPROACH

3.1 Object-Oriented Databases: General Concepts

It is very difficult to establish general characteristic of object oriented databases. One of the reasons is that even though Atkinson et al. [1] proposed some rules to unify OODB concepts, there is still not agreement among researchers even in the terminology used. In summarizing the approaches from different sources, the general characteristic of object-oriented database are as follows:

- The object represents "everything" ([8], p.635). The more precise definition given by Özsu and Valduriez [30] is that the object is the pair of values (id, v), where id is a system created identifier and v is the value that object can have. Other authors [9, 26] define an object as a triple (identifier, type, value), where the type represents one of the system defined structures (atomic, set, tuple, etc). Some authors [20, 19] use the term state instead of term value.
- Each object has a system defined identifier.
- Each object has a private memory and a set of operations (methods). The private memory has a value of the attributes [26]. Khoshofian names these attributes as instance variables [19].

  There are two approaches regarding the existence of the objects and values in an object-oriented database system:

  - In the approach used by Smalltalk [11], each value is represented by an object, which has its own identifier and can be accessed only by invoking the corresponding method.
  - The other approach allows the object-oriented databases to have values and objects [9, 24, 19, 16].

- The encapsulation and data hiding gives the possibility of defining the operations or functions for the specific object to manipulate it or return part of its state.

- The use of type constructor allows one to create objects of arbitrary complexity. The basic type constructors are atom, tuple and set. Also, list, array and bag are other commonly used types [9]. According to Özsu and Valduriez [30], the object can be named depending on the kind of value that it has: atomic object, tuple object, set object, and so forth. Also, they define a complex object as an object which value is represented by the identifier of another object.

- The value of the object depends on its type:
  - if the object is an atom its value is an element (without subparts) of the domain defined by the user or system.
  - if the object is a set, then its value is a set of the different object's identifiers.
  - if the object is a tuple, then its value is of the form (at_1:id_1, at_2:id_2, ..., at_n:id_n), where at_i means attribute name, and id_i means distinct object identifier.

- The class hierarchies and inheritance can be used.

The designer can also group classes into superclasses (bottom-up object-oriented approach) or divide classes in subclasses (top-down object-oriented approach). Using the terminology of a relational approach we can say that superclass is the generalization of classes or the subclass are specialization of classes [20]. If the designer defines the subclass, it is necessary that "every object of subclass must also be a member of a superclass" ([9], p.667). This means the objects of subclass inherit attributes and methods from its superclasses [20]. This allows one to draw the inheritance as a directed acyclic graph (DAG). Also, it is possible that some classes inherit methods or attributes from several different classes (multiple inheritance).

3.2 Fragmentation in Distributed Object-Oriented Databases

In the relational approach the fragmentation deals with algorithms that apply to attribute distribution. Depending on the kind of fragmentation, the information about transaction frequency, attribute usage, or predicate type is needed. According to Özsu and Valduriez [30], we need four categories of information to achieve the optimal design:

1. Database information that includes the global conceptual schema.
2. Application information: used predicates (for horizontal fragmentation). or attribute usage matrix and transaction frequencies (for vertical fragmentation).
3. Communication network information.
4. Computer system information.

The last two items are mostly used in allocation models.

However, in the object-oriented approach the conceptual schema can be more complex and apart from the features presented in a relational approach, aditional aspects have to be considered for fragmentation such as methods, hierarchical structure, and complex attributes.

These new features increase the number of possible approaches for fragmentation. For example, partitioning of classes may involve

- Finding some kind of "affinity" between classes (all defined objects of the class represent an entity here; we do not fragment the attributes and methods). "Affinity" between classes can be found in a similar way as is done for attributes in relational approach.
- "Flattening" classes and objects and finding the "affinity" between attributes.
- Distributing of the methods using the frequencies of accessing them by specific transactions followed by the fragmentation of corresponding attributes that these methods access.
- Using the frequencies of th attribute accesses for partitioning of objects of the class followed by appropriate distribution of methods.
- Following the hierarchical structure of the classes.

3.3 Related Work

In developing algorithms for horizontal partitioning, Ezeife and Barker [10] analyzed four separate cases: simple attributes and simple methods, simple attributes and complex methods, complex attributes and simple methods and complex attributes and complex methods.

They establish several new concepts such as "object pointer join" "instance object join" (p.6), "object affinity", "affinity between derived fragments" (p.12), and so forth. They also choose a class as an entity of fragmentation. In dealing with the primary and derived fragmentation their algorithms have several steps including the definition of the link graph for the classes, as well as the primary horizontal fragment and the derived horizontal fragments. Finally, they combine the primary and derived fragments, while also finding the best placement for the objects.

Karlapalem and Li [17] try to establish some common schemes for the fragmentation in object-oriented databases. They do not present specific algorithms for fragmentation, but propose some initial steps for each fragmentation scheme. They categorize three kind of classes: value-based, object-based and mixed. Additionally, they analyze the vertical, horizontal, and path partitioning schemes. For the horizontal partitioning in object-based classes, they distinguish the case of derived horizontal partitioning (when classes are fragmented based on the value of object in hierarchy) and associated horizontal partitioning (when the classes are fragmented because of their natural division into subclasses). Even though each of these schemes are described very clearly and could be used, the internal representation of the fragments that they propose may be too complicated for large object-oriented databases specifically with several levels of hierarchy.

Gruber and Amsaleg [12] propose special object grouping based on the relevance of the link. Specifically, they analyze the hierarchy links and links between different objects (when the instance variable of one object is pointing to another object) and, based on this information, they develop the object clustering that is used in the system EOS. They consider only the necessity of putting parent and child objects together. They do not analyze the possibility of the different usage of the classes and their subclasses.

Karlapalem et al. [18] describe some of the issues to be considered in an object-oriented distributed environment such as used data model, the method invocation important for allocation purposes, types of transparency, closeness of methods and attributes, inheritance hierarchy, and so forth. They do not present any algorithms for vertical partitioning, and affirm that the algorithm proposed in Navathe et al.

[29] can be used for the simple method and value-based attributes. For the complex methods and value-based attributes they propose to "flatten" objects and apply to obtained schema the concept developed in [31] for fragmentation of relation with overlapping views.

CHAPTER 4
MODELING OBJECT HIERARCHY FOR PARTITIONING

4.1 Motivation for Developing a New Tool for Vertical Partitioning in DOODB

To the best of our knowledge, only fragmentation schemes presented by Ezeife and Barker [10], and Gruber and Amsaleg [12] were implemented. Also, none of the presented algorithms consider the relationship between relational and object-oriented approaches and the possibility of extending existing algorithms from relational to object-oriented databases. Even though Karlapalem et al. [18] propose the feasibility of using the fragmentation schemes existing in relational approach in object-oriented environment, they only describe this adoption in very general form.

Moreover, the attribute usage matrix used in vertical partitioning (in all the previous works) in relational databases only consider the case when the frequencies of accessing the attributes by some specific transaction are the same for all attributes. However, in the case of object-oriented databases, the attribute usage matrix can have different frequencies of the attributes for the same transaction. For example, in the case of complex attributes, when the object to which this attribute is pointing to is accessed "indirectly" through this complex attribute, it can be accessed "directly" from some other transaction. Therefore, the Partition Evaluator proposed by Chakravarthy et al. [4] cannot be applied without modifications to the case of an object-oriented database.

In addition, not only the frequencies of accessing attributes by transactions, but also the existence and frequency of used methods should be considered in the new fragmentation scheme. Therefore, matrices that represent this quantitative information should be developed.

If we can show that it is possible to generalize algorithms and concepts developed for a relational database and use them in an object-oriented databases, we can inherit earlier research about the development of relational databases without the necessity to construct new tools from the beginning.

4.2 Assumptions

In the following we consider:

1. Object-based and no values-based approach is applied.
2. The information of the classes including the methods and attributes used by methods is given.
3. The access to the attributes is done only through the methods following the principle of information hiding, encapsulation, and abstract data. This assumption facilitates the path expression allowing one to have in it no more than two elements.
4. The information of accessing the method by specific transaction is given.
5. The frequencies of accessing transactions is specified.
6. The existence of the type tuple, set, and list are ignored.

4.3 Transaction, Method and Attribute Usage Matrices for DOODB

The information needed for partitioning can be represented in the form of the Transaction-Method Usage Matrix (TMUM), Method-Method Usage Matrix (MMUM) and Method-Attribute Usage Matrix (MAUM).

- Transaction-Method Usage Matrix (TMUM) is the matrix of frequencies that indicates if the transaction calls specific methods. Used values are zero or one that represent no call or call of the specific method respectively.

$$\begin{array}{lcccc} trans \setminus methods & m^1_{i,j} & m^2_{i,j} & \ldots & m^m_{i,j} \\ & & & & \\ tr_1 & fm_{1,1} & fm_{1,2} & \ldots & fm_{1,m} \\ tr_2 & fm_{2,1} & fm_{2,2} & \ldots & fm_{2,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ tr_T & fm_{T,1} & fm_{T,2} & \ldots & fm_{T,m} \end{array} \tag{4.1}$$

where

m	Total number of methods in the system that is being partitioned.
$m^l_{i,j}$	Method j of class i, for l=1 to m.
T	Total number of transactions that are under consideration.
tr_p	Transaction p, for p=1 to T.
$fm_{p,r}$	Frequency that transaction p accesses method r (equal to 0 or 1).

- Method-Method Usage Matrix (MMUM) is the matrix of frequencies that indicates if the method calls other methods (in the case of simple methods this matrix does not exist). Similar to the TMUM the values of zero or one represent the existence of nested calls. The value equal to two is presented only in the case when the method is called through the complex attribute.

$$\begin{array}{lcccc} methods \setminus methods & m^1_{i,j} & m^2_{i,j} & \ldots & m^m_{i,j} \\ & & & & \\ m^1_{i,j} & 1 & fmm_{1,2} & \ldots & fmm_{1,m} \\ m^2_{i,j} & fmm_{2,1} & 1 & \ldots & fmm_{2,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m^m_{i,j} & fmm_{m,1} & fmm_{m,2} & \ldots & 1 \end{array} \tag{4.2}$$

where

m	Total number of methods in the system that is being partitioned.
$m^l_{i,j}$	Method j of class i, for l=1 to m.
$fmm_{s,r}$	Frequency that method s calls method r (equal to 0, 1, or 2) .

- Method-Attribute Usage Matrix (MAUM) is the matrix that represents the number of invocations of specific attribute in one execution of a method. There are possible three values:
  - Zero - indicates that a method does not access an attribute.
  - One - indicates that a method reads the value of an atributes (retrieve).
  - Two - indicates that a method reads and writes the value of an atrtibute (update).

$$\begin{array}{lcccc} methods \setminus attr & at^1_{i,j} & at^2_{i,j} & \dots & at^n_{i,j} \\ m^1_{i,j} & fat_{1,1} & fat_{1,2} & \dots & fat_{1,n} \\ m^2_{i,j} & fat_{2,1} & fat_{2,2} & \dots & fat_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m^m_{i,j} & fat_{m,1} & fat_{m,2} & \dots & fat_{m,n} \end{array} \tag{4.3}$$

where

n	Total number of attributes in the system that is being partitioned.
$at^p_{i,j}$	Attribute j of class i, for p=1 to n.
$fat_{k,l}$	Frequency of accessing of attribute l by method k (equal to 0, 1, or 2).

The information specified above can be given by the designer of the system.

Additionally, in the same way as was done for relational approach, the frequency of using the specific transaction by applications is given:

Frequencies of used transaction by applications (ftr):

$$\begin{array}{llllll} transaction & tr_1 & tr_2 & tr_3 & \ldots & tr_T \\ frequency & ftr_1 & ftr_2 & ftr_3 & \ldots & ftr_T \end{array} \tag{4.4}$$

With TMUM, that contains the information of methods used by a specific transaction, and MAUM, that contains the frequencies of used attributes by a specific method, we can multiply them and obtain the matrix that represents the frequencies of used attributes by a specific transaction. These frequencies serve as a base for the fragmentation scheme and correspond to the Attribute Usage Matrix (AUM) in the relational approach. Why not fragment methods and then assign the attributes to the adequate fragments? The reason is obvious. The methods as a piece of software can be easily copied, or duplicated without any problems (space, concurrency control, and so forth). The adequate distribution of the data, which optimizes the store space (no unnecessary duplication) and gives good performance regarding the used transaction, is the main goal of the design in distributed databases systems, either relational or object-oriented.

Even though the obtained matrix correspond to the AUM, the method used for fragmentation in relational approach cannot be used for this object-oriented approach for the reasons explained before.

To consider our approach feasible, based on Chakravarthy et al. [4], we develop the Partition Evaluator for Object-Oriented approach (PEOO) that evaluates the partitioning scheme using the AUM, which can have different values of frequencies of attributes used by the same transaction. We apply this PEOO for exhaustive search in the same way that mentioned authors applied for relational database [4].

Moreover, our algorithm allows the user to define desired granularity of fragmentation that can vary in each class or can be the same for the system to be fragmented. These granularity levels are described as level 1 (root class), level 2 (root class and one level of subclasses), and so forth.

4.4 Description of Modeling Object Hierarchy

In the process of developing the fragmentation model, we consider the same four cases that were considered in [10].

4.4.1 Simple attribute and simple methods

In this category we have only the simple hierarchical structure, where the methods can use attributes of their own class or their superclasses. We do not have nested methods or objects as an attribute type.

In proposed algorithm we should:

1. Specify in Transaction-Method Usage Matrix (TMUM) and Method-Attribute Usage Matrix (MAUM) all used methods and attributes from the root to the established level of granularity i. If the classes of the levels greater than i exist, then for each of the classes represent the frequencies of accessing the attributes and methods as a sum of corresponding values. Note here, that if the class has subclasses, the sum should include all frequencies of methods and attributes of this class and all of its subclasses.

2. Multiply each row of the TMUM with the corresponding values of transaction frequencies (matrix 4.4).

3. Multiply TMUM and MAUM.

4. Apply the exhaustive search and PEOO for selecting the best fragmentation scheme. This search will consider the attributes of the subclasses of level greater than i as indivisible.

5. Propagate the obtained fragmentation.

Consider the following example:

$$\begin{tikzcd}
\begin{array}{c} Class_1 \\ at^1_{1,1} \quad at^2_{1,2} \\ m^1_{1,1} \quad m^2_{1,2} \end{array} \arrow[d] \arrow[dr] & & \begin{array}{c} Class_4 \\ at^7_{4,1} \quad at^8_{4,2} \\ m^7_{4,1} \quad m^8_{4,2} \end{array} \arrow[d] \arrow[dr] & \\
\begin{array}{c} Class_2 \\ at^3_{2,1} \quad at^4_{2,2} \\ m^3_{2,1} \quad m^4_{2,2} \end{array} & \begin{array}{c} Class_3 \\ at^5_{3,1} \quad at^6_{3,2} \\ m^5_{3,1} \quad m^6_{3,2} \end{array} & \begin{array}{c} Class_5 \\ at^9_{5,1} \quad at^{10}_{5,2} \\ m^9_{5,1} \quad m^{10}_{5,2} \end{array} \arrow[d] & \begin{array}{c} Class_6 \\ at^{11}_{6,1} \quad at^{12}_{6,2} \\ m^{11}_{6,1} \quad m^{12}_{6,2} \end{array} \\
 & & \begin{array}{c} Class_7 \\ at^{13}_{7,1} \quad at^{14}_{7,2} \\ m^{13}_{7,1} \quad m^{14}_{7,2} \end{array} &
\end{tikzcd}$$

To make this graph easier to understand we avoid repetition of methods and attributes to the subclasses, assuming that all methods and attributes of superclasses can be accessed by their subclasses. Here, if the level of granularity is 1 for both root classes ($Class_1$ and $Class_4$), we will have TMUM in the form:

$trans \setminus meth$	$m^1_{1,1}$	$m^2_{1,2}$	$m^7_{4,1}$	$m^8_{4,2}$	M_2	M_3	M_5	M_6
tr_1	$fm_{1,1}$	$fm_{1,2}$	$fm_{1,3}$	$fm_{1,4}$	$fm_{1,5}$	$fm_{1,6}$	$fm_{1,7}$	$fm_{1,8}$
tr_2	$fm_{2,1}$	$fm_{2,2}$	$fm_{2,3}$	$fm_{2,4}$	$fm_{2,5}$	$fm_{2,6}$	$fm_{2,7}$	$fm_{2,8}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
tr_T	$fm_{T,1}$	$fm_{T,2}$	$fm_{T,3}$	$fm_{T,4}$	$fm_{T,5}$	$fm_{T,6}$	$fm_{T,7}$	$fm_{T,8}$

The values in the columns M_i represent the sum of the frequencies of accessing the methods by corresponding transaction. For example, the value $fm_{1,5}$ is the sum of frequencies of accessing the methods $m^3_{2,1}$ and $m^4_{2,2}$ of $Class_2$ by transaction tr_1. For the $Class_5$ the corresponding values include not only the frequencies of accessing methods of this class, but also the frequencies of accessing the methods of its subclass: $Class_7$.

The MAUM is represented as follows:

$$\begin{array}{lcccccccc}
meth \setminus attr & at^{1}_{1,1} & at^{2}_{1,2} & at^{7}_{4,1} & at^{8}_{4,2} & AT_2 & AT_3 & AT_5 & AT_6 \\
m^{1}_{1,1} & fat_{1,1} & fat_{1,2} & fat_{1,3} & fat_{1,4} & fat_{1,5} & fat_{1,6} & fat_{1,7} & fat_{1,8} \\
m^{2}_{1,2} & fat_{2,1} & fat_{2,2} & fat_{2,3} & fat_{2,4} & fat_{2,5} & fat_{2,6} & fat_{2,7} & fat_{2,8} \\
m^{7}_{4,1} & fat_{3,1} & fat_{3,2} & fat_{3,3} & fat_{3,4} & fat_{3,5} & fat_{3,6} & fat_{3,7} & fat_{3,8} \\
m^{8}_{4,2} & fat_{4,1} & fat_{4,2} & fat_{4,3} & fat_{4,4} & fat_{4,5} & fat_{4,6} & fat_{4,7} & fat_{4,8} \\
M_2 & fat_{5,1} & fat_{5,2} & fat_{5,3} & fat_{5,4} & fat_{5,5} & fat_{5,6} & fat_{5,7} & fat_{5,8} \\
M_3 & fat_{6,1} & fat_{6,2} & fat_{6,3} & fat_{6,4} & fat_{6,5} & fat_{6,6} & fat_{6,7} & fat_{6,8} \\
M_5 & fat_{7,1} & fat_{7,2} & fat_{7,3} & fat_{7,4} & fat_{7,5} & fat_{7,6} & fat_{7,7} & fat_{7,8} \\
M_6 & fat_{8,1} & fat_{8,2} & fat_{8,3} & fat_{8,4} & fat_{8,5} & fat_{8,6} & fat_{8,7} & fat_{8,8}
\end{array}$$

In this MAUM the columns represented as AT_k are the sum of the frequencies of accessing the attributes by corresponding method and are represented in a similar way as in TMUM. Note that the intersections of the rows M_i and column AT_k have the values that represent the sum of the frequencies of all the attributes in AT_k accessed by methods in M_i.

For exhaustive search (step 4 of the previous algorithm), the attributes of the subclasses $Class_2$, $Class_3$, $Class_5$, $Class_6$ and $Class_7$ are considered indivisible. This means we will not analyze the case when the attributes of one of these classes belong to different fragments. Moreover, this consideration implies that the attributes of all subclasses presented in the same hierarchy path should form one indivisible entity (for $Class_5$ and $Class_7$ attributes $at^{9}_{5,1}$, $at^{10}_{5,2}$, $at^{13}_{7,1}$ and $at^{14}_{7,2}$ should be considered as indivisible).

If the level of granularity is two for root $Class_1$ and one for root $Class_4$ the corresponding matrices will be:

$$\begin{array}{lcccccc}
trans \setminus meth & m^{1}_{1,1} & m^{2}_{1,2} & \dots & m^{8}_{4,2} & M_5 & M_6 \\
tr_1 & fm_{1,1} & fm_{1,2} & \dots & fm_{1,8} & fm_{1,9} & fm_{1,10} \\
tr_2 & fm_{2,1} & fm_{2,2} & \dots & fm_{2,8} & fm_{2,9} & fm_{2,10} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
tr_T & fm_{T,1} & fm_{T,2} & \dots & fm_{T,8} & fm_{T,9} & fm_{T,10}
\end{array}$$

$$\begin{array}{lcccccc}
meth \setminus attr & at^1_{1,1} & at^2_{1,2} & \dots & at^8_{4,2} & AT_5 & AT_6 \\
 & & & & & & \\
m^1_{1,1} & fat_{1,1} & fat_{1,2} & \dots & fat_{1,8} & fat_{1,9} & fat_{1,10} \\
m^2_{1,2} & fat_{2,1} & fat_{2,2} & \dots & fat_{2,8} & fat_{2,9} & fat_{2,10} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
m^8_{4,2} & fat_{8,1} & fat_{8,2} & \dots & fat_{8,8} & fat_{8,9} & fat_{8,10} \\
M_5 & fat_{9,1} & fat_{9,2} & \dots & fat_{9,8} & fat_{9,9} & fat_{9,10} \\
M_6 & fat_{10,1} & fat_{10,2} & \dots & fat_{10,8} & fat_{10,9} & fat_{10,10}
\end{array}$$

The indivisible group of attributes are defined in a similar way as in the case of the granularity level one.

The step of propagation can be shown for $Class_1$. Suppose that after applying the exhaustive search with granularity level one, we obtain two fragments: the first has the attribute $at^1_{1,1}$ and the second has $at^2_{1,2}$. The propagation of this fragmentation can be presented in the following manner:

$$\begin{tikzcd}
 & & & Class_1 \arrow[dll] \arrow[drr] & & & \\
 & \begin{array}{c} Class_1' \\ at^1_{1,1} \\ m^1_{1,1} \; m^2_{1,2} \end{array} \arrow[dl] \arrow[dr] & & & & \begin{array}{c} Class_1'' \\ at^2_{1,2} \\ m^1_{1,1} \; m^2_{1,2} \end{array} \arrow[dl] \arrow[dr] & \\
\begin{array}{c} Class_2 \\ at^3_{2,1} \; at^4_{2,2} \\ m^3_{2,1} \; m^4_{2,2} \end{array} & & \begin{array}{c} Class_3 \\ at^5_{3,1} \; at^6_{3,2} \\ m^5_{3,1} \; m^6_{3,2} \end{array} & & \begin{array}{c} Class_2 \\ at^3_{2,1} \; at^4_{2,2} \\ m^3_{2,1} \; m^4_{2,2} \end{array} & & \begin{array}{c} Class_3 \\ at^5_{3,1} \; at^6_{3,2} \\ m^5_{3,1} \; m^6_{3,2} \end{array}
\end{tikzcd}$$

,

Notice that for implementation purposes the attributes and methods of $Class_2$ and $Class_3$ either can be replicated as shown above or can maintain only one copy of each subclass. Moreover, as was mentioned before, because methods are easy to duplicate, we propose their replications.

In the case of granularity level two we can represent the propagation of fragmentation in a similar way. Suppose the exhaustive search is applied to the root $Class_4$

and its subclasses $Class_5$ and $Class_6$. For the simplicity of representation we assume that the best fragmentation has two fragments with the following attributes:

fragment 1: $at^7_{4,1}$, $at^9_{5,1}$ and $at^{11}_{6,1}$.

fragment 2: $at^8_{4,2}$, $at^{10}_{5,2}$ and $at^{12}_{6,2}$.

The graphical representation of the propagation will appear as follows:

$$\begin{tikzcd}
 & & & Class_4 \arrow[dll] \arrow[drr] & & & \\
 & \begin{array}{c} Class'_4 \\ at^7_{4,1} \\ m^7_{4,1}\ \ m^8_{4,2} \end{array} \arrow[d] \arrow[drrrr] & & & & \begin{array}{c} Class''_4 \\ at^8_{4,2} \\ m^7_{4,1}\ \ m^8_{4,2} \end{array} \arrow[dllll] \arrow[d] & \\
 & Class_5 \arrow[dl] \arrow[dr] & & & & Class_6 \arrow[dl] \arrow[dr] & \\
\begin{array}{c} Class'_5 \\ at^9_{5,1} \\ m^9_{5,1}\ \ m^{10}_{5,2} \end{array} \arrow[dr] & & \begin{array}{c} Class''_5 \\ at^{10}_{5,2} \\ m^9_{5,1}\ \ m^{10}_{5,2} \end{array} \arrow[dl] & & \begin{array}{c} Class'_6 \\ at^{11}_{6,1} \\ m^{11}_{6,1}\ \ m^{12}_{6,2} \end{array} & & \begin{array}{c} Class''_6 \\ at^{12}_{6,2} \\ m^{11}_{,1}\ \ m^{12}_{,2} \end{array} \\
 & \begin{array}{c} Class_7 \\ at^{13}_{7,1}\ \ at^{14}_{7,2} \\ m^{13}_{7,1}\ \ m^{14}_{7,2} \end{array} & & & & &
\end{tikzcd}$$

Here, two fragments of $Class_5$ are pointing to the same $Class_7$ avoiding the duplication of objects of $Class_7$.

Even though this graphical representation of propagation seems very complicated for this simple class, in reality it can be implemented using ideas presented in system ORION [22], where each class has its own descriptor. Because each object has a unique object identifier (UID) that consists of a pair of values, which are class identifier and object identifier, the information in which fragment the desired attribute is presented can be maintained in the class descriptor.

4.4.2 Simple attributes and complex methods

The difference here with the previous case of simple attributes and simple methods is that we now have the possibility of nested calls either in the same class or between the classes that belong to upper levels of the same path of the hierarchy tree.

Graphically, in a one level nested call, the situation can be presented as:

$$\begin{tikzcd} tr_k \arrow[r] & m^p_{i,j} \arrow[r] \arrow[d] & m^r_{i,j} \\ & m^s_{i,j} & \end{tikzcd}$$

The edges from this graph will be presented in matrices TMUM and MMUM on the corresponding positions with the value equal to 1.
More complicated two level nested call looks as follows:

$$\begin{tikzcd} tr_k \arrow[r] & m^p_{i,j} \arrow[r] \arrow[d] & m^r_{i,j} \arrow[d] \\ & m^s_{i,j} & M_q \end{tikzcd}$$

Depending of the level of granularity, the transaction can access a method (m) or a method that is part of an indivisible group of methods (M). These nested method calls should be reflected in MMUM. The difference in the approach that was presented earlier will be that before multiplying the matrices TMUM and MAUM, we must first multiply the TMUM and MMUM to take into account all nested calls. Having modified the matrix TMUM, we can multiply it with MAUM and follows the steps 4 and 5 from the first case.

4.4.3 Complex attributes and simple methods

Complex objects are objects which have attribute(s) pointing to other object(s). In this case we have the situation that when a complex attribute is accessed in reality, we access this attribute and some method(s) of the class that this attribute is pointing to. Because of the encapsulation principle, we will not have the situation in which complex attributes access an attribute of another class directly. Now, we have not only the relationship between method and attribute (the attributes that the specific method uses presented in MAUM), but also a new relationship between the methods. This new relationship needs to be captured in the MMUM, giving us the case of complex attributes and complex methods. On the other hand, the relationship between the method and the complex attribute can be ignored, after reflecting in MMUM the relationship between method and method. This is because the complex attributes can be seen as a certain kind of "virtual" attribute. The assignment of this attribute to some specific fragment after applying the fragmentation algorithm, does not give to us the necessary information concerning how to fragment the attributes of the class that this attribute is pointing to. In our approach we propose to eliminate this "virtual" attribute from the partitioning scheme. It always can be placed and duplicated in any fragment that it is needed. Deleting this attribute from the fragmentation algorithm gives to us the case of simple attributes and nested method calls instead of complex attributes and simple method calls. For example, suppose complex attribute $at^k_{i,j}$ is used in a path with the following methods: $m^l_{i,j}i$ and $m^s_{i,j}$. This means that in some methods we can find the invocation such as: $at^k_{i,j}.m^l_{i,j}$ and $at^l_{i,j}.m^s_{i,j}$ (because of the principle of encapsulation the invocation of complex attribute cannot be found directly in the transaction). This is how it looks when represented graphically:

$$\begin{tikzcd}
tr_k \arrow[r] & m^p_{i,j} \arrow[r] & at^k_{i,j} \arrow[r] \arrow[d] & m^l_{i,j} \\
 & & m^s_{i,j} &
\end{tikzcd}$$

If some transaction k calls the method that uses this complex attribute, we propose to eliminate this attribute from the graph. This results in the situation that the methods $m^l_{i,j}$ and $m^s_{i,j}$ are called in the method $m^p_{i,j}$. The relationship between methods are already reflected in MMUM in the positions (p,l) and (p,s) and no modifications need be done.

In case when the access through a complex attribute is more complicated, because some method that this complex attribute is pointing to accesses another complex attribute, we make a modification to the graph in the same way as shown in the previous example.

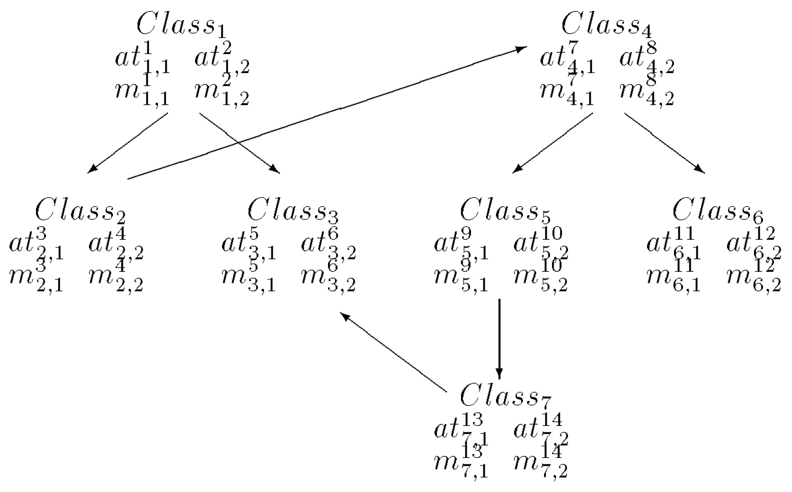
For example, if we have the following situation:

$$\begin{tikzcd}
tr_k \arrow[r] & m^p_{i,j} \arrow[r] & at^k_{i,j} \arrow[r] \arrow[d] & m^l_{i,j} \arrow[d] & \\
 & & m^s_{i,j} & at^r_{i,j} \arrow[r] \arrow[d] & m^w_{i,j} \\
 & & & m^v_{i,j} &
\end{tikzcd}$$

the graph can be rewritten as:

$$\begin{tikzcd}
tr_k \arrow[r] \arrow[d] & m^l_{i,j} \arrow[r] \arrow[d] & m^w_{i,j} \\
m^s_{i,j} & m^v_{i,j} &
\end{tikzcd}$$

Also, we can consider different levels of granularity in a similar way as was done for simple attributes and simple methods. Consider the following object hierarchy:


$Class_1$ $at^1_{1,1}$ $at^2_{1,2}$ $m^1_{1,1}$ $m^2_{1,2}$
$Class_4$ $at^7_{4,1}$ $at^8_{4,2}$ $m^7_{4,1}$ $m^8_{4,2}$
$Class_2$ $at^3_{2,1}$ $at^4_{2,2}$ $m^3_{2,1}$ $m^4_{2,2}$
$Class_3$ $at^5_{3,1}$ $at^6_{3,2}$ $m^5_{3,1}$ $m^6_{3,2}$
$Class_5$ $at^9_{5,1}$ $at^{10}_{5,2}$ $m^9_{5,1}$ $m^{10}_{5,2}$
$Class_6$ $at^{11}_{6,1}$ $at^{12}_{6,2}$ $m^{11}_{6,1}$ $m^{12}_{6,2}$
$Class_7$ $at^{13}_{7,1}$ $at^{14}_{7,2}$ $m^{13}_{7,1}$ $m^{14}_{7,2}$

If the granularity level is one for both root classes the methods $m^1_{1,1}$, $m^2_{1,2}$, $m^7_{4,1}$ and $m^8_{4,2}$ will be represented in the corresponding matrices. For the classes $Class_2$, $Class_3$, $Class_5$ together with $Class_7$ and $Class_6$ the frequencies of accessing methods will be presented as summations of the frequencies of the methods of the above mentioned classes. In other words, based on the explanation given for simple attributes and simple methods, in addition to the specified above methods, we will have the following groups of methods:

- M_2 contains methods of $Class_2$.
- M_3 contains methods of $Class_3$.
- M_5 contains methods of $Class_5$ and $Class_7$.
- M_6 contains methods of $Class_6$.

The same consideration will be applied to the attributes.

After preparing TMUM, MMUM (if needed) and MAUM, applying an exhaustive search to the specified granularity level(s), and obtaining fragmentation of attributes, the question that is left is how to propagate the fragmentation. The way to do it is exactly the same as in the first case of simple attributes and simple methods. Starting from the root for each fragment we will analyze how the attributes of each class have been fragmented. Then, we will propagate this fragmentation to the subclasses. In each subclass we first reflect the inherited fragmentation and then we analyze the proper attributes of this subclass. Finally, we apply the fragmentation to these proper attributes in the same way as was applied to the root (see second example of the first case of simple methods and simple attributes, p.24). This process is repeated recursively until granularity level i is reached. For classes of the levels greater than i, we apply the concept of the indivisible group of attributes, which was explained earlier.

4.4.4 Complex attributes and complex methods

We can see that the case of complex attributes and complex methods has been included in the analysis of the previous case.

For all the cases, the applied fragmentation algorithm is the same as presented for simple methods and simple attributes except for the following refinement in the case of nested calls:
If the nested calls exist, we should prepare MMUM and multiply it with the TMUM before the second step.

CHAPTER 5
DEVELOPMENT OF AN OBJECTIVE FUNCTION

5.1 Need for an Objective Function

As mentioned previously, most algorithms for vertical partitioning in a relational database use affinity usage matrix as an input. This matrix cannot express the "affinity" between more than two attributes. Hence, the tool that uses attribute usage matrix can accurately reflect the real behavior of the system. Moreover, some existing algorithms declare themselves as the best and give different "optimal" fragmentation for the same input information. The result of one algorithm very often cannot be compared to the results of others. Therefore, the necessity of having a tool that can measure the "goodness" of presenting results is obvious. However, this tool was developed for a relational database after many fragmentation algorithms were presented. Having such a tool for an object-oriented database, at an early stages of DOODB research, allows the designer to evaluate the new partitioning algorithms as they are becoming available.

The goal of partitioning the attributes is to obtain the minimum processing cost for a given set of transactions and their usage of attributes. It is unlikly to achive an ideal fragmentation scheme, where any transaction locally accesses only the attributes that are needed and does not need to perform any remote accesses. The objective function proposed here tries to balance the cost of local and remote accesses for a given transactions. According to Chakravarthy et al. [4] the overall transaction processing cost in a distributed environment consists of two elements:

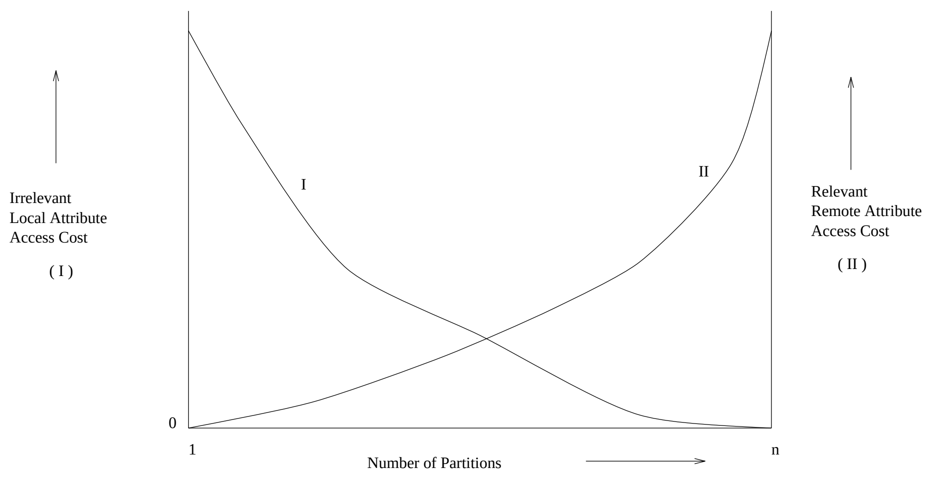


Figure 5.1. Behavior of partition evaluator in relational approach

1. The first component, called irrelevant local attribute access cost, represents the cost of accessing irrelevant attribute when accessing the object in the local site, assuming that all data fragments required by a transaction are available locally

2. The second component, called relevant remote attribute cost, includes the cost of accessing the relevant attributes required by the transaction from remote sites.

Chakravarthy et al. [4] showed the expected behavior of the objective function using the above components. It is shown in the graph presented in the fig. 5.1.

Individually, the irrelevant local attribute cost should have a maximum value for one fragment and equal to zero for a partition size (i.e. number of fragments in the partition) equal to the number of attributes. The relevant remote access cost should be zero and maximum values for these two extremes respectively. Moreover, as stated by the previously mentioned authors, the local cost should be more responsive to smaller partition sizes in contrary to the remote cost that should be more responsive to larger fragments in between these two extreme values.

As is evident, the partitioning evaluator for an object-oriented database should have the same behavior as the partitioning evaluator presented above for a relational database. However, the irrelevant local cost will have one additional component. Because the frequencies of attributes for the same transaction can be different, the penalty for accessing some relevant attributes more than others will be considered as a penalty for the local cost.

5.2 Assumptions and Notations

We mentioned before that the difference between the original AUM and the TAUM developed for object oriented environment is the fact that the AUM in a relational approach uses the frequencies of a transaction equal for all attributes, and the new matrix can have different values of frequency for each attribute used by the transaction. This means, the matrix (5.1), which is presented for a relational database, should be replaced with the matrix (5.2), developed for an object oriented approach:

$$\begin{array}{lcccc} trans \setminus attrs & at_j & at_i & \dots & at_i \\ & & & & \\ tr_1 & q_1 & q_1 & \dots & q_1 \\ tr_2 & q_2 & q_2 & \dots & q_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ tr_T & q_T & q_T & \dots & q_T \end{array} \tag{5.1}$$

where q_t represents the frequency of transaction t=1,2,...,T.

$$\begin{array}{lcccc} trans \setminus method & at^1_{i,j} & at^2_{i,j} & \dots & at^n_{i,j} \\ & & & & \\ tr_1 & f_{1,1} & f_{1,2} & \dots & f_{1,n} \\ tr_2 & f_{2,1} & f_{2,2} & \dots & f_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ tr_T & f_{T,1} & f_{T,2} & \dots & f_{T,n} \end{array} \tag{5.2}$$

Even though it is possible to replicate data, in this first step of developing PEOO we assume no replication scheme. Also, we assume that the fragmentation phase is followed by fragment allocations. In addition, non-overlapping attributes within the fragments are considered. This means n attributes are partitioned into M fragments $(P_1, P_2, \dots, P_M)$, with n_i attributes in each fragment. Thus $\sum_{i=1}^{M} n_i = n$. These assumptions allow the reader and author of this thesis to simplify the steps of the development of PEOO without losing its generality. In addition, for simplicity we assume that single access to a data in a local fragment corresponds to a unit of cost.

Since we do not know how the data is allocated during partitioning, the cost of relevant remote attributes is calculated assuming that the data fragments needed by the transaction are allocated in different sites. Due to lack of the specific information about network, the real transmission cost is ignored taking a unit cost for each access to the remote fragment. It is important to notify that the contribution to the remote access cost due to irrelevant attributes is already included in the first term.

We do not make any assumptions about the input data.

Moreover, the following, are the parameters proposed in [4] and also used here with the small modifications:

n	Total number of attributes in a relation that is being partitioned.
T	Total number of transactions that are under consideration.
q_t	Frequency of transaction t for $t = 1, 2, \ldots, T$.
M	Total number of fragments of a partition.
n_i	Number of attributes in fragment i.
n^r_{ikt}	Total number of attributes that are in fragment k accessed remotely with respect to fragment i by transaction t.
f^i_{tj}	Frequency of transaction t accessing attribute j in fragment i note that f^i_{tj} was either 0 or q_t for relational approach
A_{ij}	Attribute Vector for attribute j in fragment i. t-th component of this vector is f^i_{tj}
S_{it}	Set of relevant attributes in fragment i that the transaction t accesses; it is empty if t does not need fragment i.
$\|S_{it}\|$	Number of relevant attributes in fragment i that the transaction t

	accesses.
I_{it}	Set of irrelevant attributes in fragment i that the transaction t accesses; it is empty if t needs all attributes from fragment i.
$\|I_{it}\|$	Number of irrelevant attributes in fragment i that transaction t accesses.
R_{itk}	Set of relevant attributes in fragment k accessed remotely with respect to fragment i by transaction t; these are attributes not in fragment i but needed by t
$\|R_{itk}\|$	Number of relevant attributes in fragment k accessed remotely with respect to fragment i by transaction t

5.3 Local Access Cost

For this component we will use the same square-error criterion as was used by Chakravarthy et al. [4] based on algorithm developed by Jain and Dubes [15]. This criterion gives the penalty factor each time an irrelevant attribute is accessed. The best local processing cost is when this value is minimum.

The Attribute Vector introduced by Chakravarthy et al. [4] is as follows:

$$A_{ij} = \begin{bmatrix} f^i_{1j} \\ f^i_{2j} \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ f^i_{tj} \end{bmatrix}$$

where f^i_{tj} is the frequency of the transaction t accessing attribute j in fragment i. Note that Chakravarthy et al. [4] presented f^i_{tj} as either 0 or q_t.

The mean vector V_i for fragment i, according to previous assumptions for non-overlapping fragments, can be defined as follows:

$$V_i = \begin{bmatrix} \frac{\sum_{j=1}^{|S_{i1}|} f_{1j}^i}{n_i} \\ \frac{\sum_{j=1}^{|S_{i2}|} f_{2j}^i}{n_i} \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \frac{\sum_{j=1}^{|S_{it}|} f_{tj}^i}{n_i} \end{bmatrix}$$

Here $|S_{it}|$ is the number of attributes in partition i that the transaction t accesses and n_i is the number of attributes in partition i. (The summation should have the upper limit equal to n_i. However the f_{tj}^i is equal to 0 for all irrelevant attributes and we can use the value of $|S_{it}|$ instead of n_i).

The square-error e_i^2 for the fragment P_i and square-error for the entire partition scheme E_L^2 are defined in the same way as for a relational approach:

$$e_i^2 = \sum_{j=1}^{n_i} (A_{ij} - V_i)^T (A_{ij} - V_i) \tag{5.3}$$

$$E_L^2 = \sum_{i=1}^{M} e_i^2 \tag{5.4}$$

The development of PE for a relational approach showed the penalty factor for accessing irrelevant local attributes. However, the PEOO will have two components:

1. It shows the penalty for accessing the irrelevant attributes (similar to the relational approach).

2. It shows the penalty for accessing the relevant attributes. This component is due the fact that the frequencies of accessing attributes by a specific transaction

are different. Thus, we should pay a penalty for accessing attributes more times than is necessary.

Rewriting the E_L^2 differently we will show the contribution of these two components.

Substituting V_i and A_{ij} in equation (5.4) for E_L^2, we obtain

$$E_L^2 = \sum_{i=1}^{M}\sum_{j=1}^{n_i}\left[f_{1j}^i - \frac{\sum_{p=1}^{|S_{i1}|} f_{1p}^i}{n_i}, \ldots, f_{tj}^i - \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i} \right] \begin{bmatrix} f_{1j}^i - \frac{\sum_{p=1}^{|S_{i1}|} f_{1p}^i}{n_i} \\ f_{2j}^i - \frac{\sum_{p=1}^{|S_{i2}|} f_{2p}^i}{n_i} \\ \ldots \\ \ldots \\ \ldots \\ \ldots \\ f_{tj}^i - \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i} \end{bmatrix} \tag{5.5}$$

This formula can be reduced to

$$= \sum_{i=1}^{M}\sum_{j=1}^{n_i}\sum_{t=1}^{T}\left[\left(f_{tj}^i - \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i} \right) * \left(f_{tj}^i - \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i} \right) \right] \tag{5.6}$$

After expanding the inside terms, we come up with

$$= \sum_{i=1}^{M}\sum_{j=1}^{n_i}\sum_{t=1}^{T}\left[\left(f_{tj}^i\right)^2 + \left(\frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i} \right)^2 - 2 * \left(f_{tj}^i * \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i} \right) \right] \tag{5.7}$$

Rewriting the above in a different way, we obtain

$$E_L^2 = \sum_{i=1}^{M}\sum_{t=1}^{T}\left[\sum_{j=1}^{n_i}\left(f_{tj}^i\right)^2 + \sum_{j=1}^{n_i}\left(\frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)^2 - 2*\sum_{j=1}^{n_i}\left(f_{tj}^i * \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)\right] \tag{5.8}$$

Now, we can divide the $\sum_{j=1}^{n_i}$ into two parts: $\sum_{j=1}^{|S_{it}|}$ and $\sum_{j=1}^{|I_{it}|}$, where $I_{it} = n_i - S_{it}$.

$$\begin{aligned} E_L^2 &= \sum_{i=1}^{M}\sum_{t=1}^{T}\left[\sum_{j=1}^{|S_{it}|}\left(f_{tj}^i\right)^2 + \sum_{j=1}^{|I_{it}|}\left(f_{tj}^i\right)^2 + \sum_{j=1}^{|S_{it}|}\left(\frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)^2 + \sum_{j=1}^{|I_{it}|}\left(\frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)^2\right. \\ &\quad \left. - 2*\sum_{j=1}^{|S_{it}|}\left(f_{tj}^i * \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right) - 2*\sum_{j=1}^{|I_{it}|}\left(f_{tj}^i * \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)\right] \end{aligned} \tag{5.9}$$

Grouping corresponding terms for relevant and irrelevant attributes, we obtain

$$\begin{aligned} E_L^2 &= \sum_{i=1}^{M}\sum_{t=1}^{T}\left[\sum_{j=1}^{|S_{it}|}\left(f_{tj}^i\right)^2 + \sum_{j=1}^{|S_{it}|}\left(\frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)^2 - 2*\sum_{j=1}^{|S_{it}|}\left(f_{tj}^i * \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)\right. \\ &\quad \left. + \sum_{j=1}^{|I_{it}|}\left(f_{tj}^i\right)^2 + \sum_{j=1}^{|I_{it}|}\left(\frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)^2 - 2*\sum_{j=1}^{|I_{it}|}\left(f_{tj}^i * \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)\right] \end{aligned} \tag{5.10}$$

That gives us

$$E_L^2 = \sum_{i=1}^{M}\sum_{t=1}^{T}\left[\sum_{j=1}^{|S_{it}|}\left(f_{tj}^i - \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)^2 + \sum_{j=1}^{|I_{it}|}\left(f_{tj}^i - \frac{\sum_{p=1}^{|S_{it}|} f_{tp}^i}{n_i}\right)^2\right] \tag{5.11}$$

The first term represents the penalty cost for accessing relevant attributes that have different frequencies. The second term represents the penalty cost for accessing irrelevant attributes.

We can easily transform this formula to the relational case. We will have $f^i_{tj} = q_t$ for relevant attributes and $f^i_{tj} = 0$ for irrelevant attributes. Also $\sum_{j=1}^{|S_{it}|}$ is equal to the value of $|S_{it}|$ and $\sum_{j=1}^{|I_{it}|}$ is equal to value of $n_i - |S_{it}|$. That gives us:

$$E_L^2 = \sum_{i=1}^{M}\sum_{t=1}^{T}\left[|S_{it}| * q_t^2\left(1 - \frac{|S_{it}|}{n_i}\right)^2 + (n_i - |S_{it}|)\left(q_t * \frac{|S_{it}|}{n_i}\right)^2\right] \tag{5.12}$$

which is the same inter-media formula as presented by Chakravarthy et al. [4]. that can be transformed to its final form giving a penalty cost for accessing irrelevant attributes in a relational approach:

$$E_L^2 = \sum_{i=1}^{M}\sum_{t=1}^{T}\left[q_t^2 * |S_{it}|\left(1 - \frac{|S_{it}|}{n_i}\right)\right] \tag{5.13}$$

Therefore, we can see that the formula for local access cost for relational approach is a special case of the formula presented above for the object-oriented approach. In fact, the new formula can be used for the relational model as well when the transaction frequencies are not the same for all attributes.

5.4 Remote Access Cost

Based on the explanation given by Chakravarthy et al. [4] and the previous part of this thesis the formula for the penalty cost of accessing remote attributes is calculated as follows: given a set of partitions, for each transaction running on the partition compute the ratio of the number of remote attributes to be accessed to the total number of attributes in each of the remote partitions. Here, in contrast to the formula presented for a relational approach, the individual frequencies for accessing each attribute are used, because of the possibility of having different frequencies of accessing the attributes by the same transaction.

$$E_R^2 = \sum_{t=1}^{T} \Delta_{i=1}^{M} \sum_{k \neq i} \left[\sum_{p_k} (f_{tp_k}^k)^2 * \frac{|R_{itk}|}{n_{itk}^{rem}} \right] \tag{5.14}$$

where p_k indicates relevant attributes in the fragment k accessed remotely with respect to the fragment i by the transaction t. Similar to the relational PE, Δ is an operator for the minimum, maximum, or average over all i and represents optimistic, pessimistic, and average estimates of remote access cost.

This formula is a generalization of the formula presented by Chakravarthy et al. [4] for relational distributed databases. With the suppositions $\sum_{p_k} = |R_{itk}|$ and $f_{tp_k} = q_k$ concerning the relational approach, we obtain the same formula for Remote Access Cost as presented in the previously mentioned paper.

Our Partition Evaluator function for an object-oriented approach (PEOO) is given by

$$PEOO = E_L^2 + E_R^2 \tag{5.15}$$

Based on the formulas (5.11) and (5.14) and on the analysis similar to presented by Chakravarthy et al. [4], the expected behavior of this partition evaluator is equal to the one shown in the fig 5.1.

CHAPTER 6
IMPLEMENTATION AND RESULTS OF THE EXPERIMENTS

6.1 Implementation

To analyze the behavior of PEOO several tests were developed. According to the previous explanation the same behavior as presented by Chakravarthy et al. [4] regarding to local and remote costs was expected.

The cases of simple attributes and simple methods, and complex attributes and complex methods were tested. The case of simple attributes and complex methods gives only the additional matrix multiplication (TMUM and MMUM). We belive that experiments representing this case do not give additional information and are hard to follow at the theoretical level. As stated before, the case of complex attributes and simple methods can be presented as a case of simple attributes and complex methods. Therefore, some examples are presented to confirm this hypothesis.

In addition, the number of classes and subclasses used in the experiments were limited to allow the readers to understand the process of fragmentation. Moreover, because exhaustive enumeration of possible attribute fragmentation was used, the limit of a total of eight attributes was established (except the examples with different granularity levels). With a more complicated hierarchical structure and a complex schema of method calls, the theoretical analysis of expected results will be uncertain. The basic idea was to show the behavior of PEOO and compare the best fragmentation obtained from a theoretical analysis and a program run. The program written in C++ that calculates the values of local and remote costs for an object-oriented approach was run for all the possible combinations (4140) of attributes with the number of fragments varying from one to eight. The PEOO was applied for each of these combinations. During the program execution, each transaction is run on each fragment. Depending on the selected option (average, minimum or maximum) the value of PEOO is calculated. The optimal value with the partitioning scheme is presented as a result of the program.

We assume, that if a transaction should be run on a fragment at least one attribute that this transaction needs is presented in this fragment. If this is not the case, the transaction is not run on this fragment.

For the case of simple attributes and simple methods the following hierarchical class representation was used:

$$\begin{tikzcd}
& \begin{array}{c} Class_1 \\ at^1_{1,1}\ at^2_{1,2}\ at^3_{1,3} \\ m^1_{1,1}\ m^2_{1,2}\ m^3_{1,3} \end{array} \arrow[dl] \arrow[dr] & \\
\begin{array}{c} Class_2 \\ at^4_{2,1}\ at^5_{2,2} \\ m^4_{2,1}\ m^5_{2,2}\ m^6_{2,3} \end{array} & & \begin{array}{c} Class_3 \\ at^6_{3,1}\ at^7_{3,2}\ at^8_{3,3} \\ m^7_{3,1}\ m^8_{3,2}\ m^9_{3,3}\ m^{10}_{3,4} \end{array}
\end{tikzcd}$$

Because the access to the attributes are done through the methods following the principles of encapsulation and information hiding, the methods of each class can only use the attributes of their own class or superclasses.

For the case of complex attributes and complex method calls the following hierarchical representation of classes was used:

$$\begin{tikzcd}
& \begin{array}{c} Class_1 \\ at^1_{1,1}\ at^2_{1,2} \\ m^1_{1,1}\ m^2_{1,2}\ m^3_{1,3} \end{array} \arrow[dl] \arrow[dr] & & \begin{array}{c} Class_4 \\ at^7_{4,1}\ at^8_{4,2} \\ m^9_{4,1}\ m^{10}_{4,2}\ m^{11}_{4,3} \end{array} \\
\begin{array}{c} Class_2 \\ at^3_{2,1}\ at^4_{2,2} \\ m^4_{2,1}\ m^5_{2,2}\ m^6_{2,3} \end{array} \arrow[urrr] & & \begin{array}{c} Class_3 \\ at^5_{3,1}\ at^6_{3,2} \\ m^7_{3,1}\ m^8_{3,2} \end{array} &
\end{tikzcd}$$

We assume the attribute $at^3_{2,1}$ to be the complex attribute that points to $Class_4$. Also, the method $m^4_{2,1}$ of the $Class_2$ calls the methods $m^9_{4,1}$ and $m^{11}_{4,3}$ of the $Class_3$ through this complex attribute $at^3_{2,1}$ and $m^5_{2,2}$ calls the method $m^{10}_{4,2}$ through the same attribute. This means, in some transactions method $m^4_{2,1}$ can have the call such as: $at^3_{2,1}.m^9_{4,1}$.

Because of a limited number of attributes that can be managed using an exhaustive partitioning algorithm, the presented classes have only two or three attributes each. Moreover, in the first part of the experiments, we assume that the desired granularity level covers all levels of classes presented in our examples. This allows us to consider all attributes for fragmentation. To see the effect of using different levels of granularity more attributes need to be used. The last three examples presented in this chapter were provided to show the feasibility of using our PEOO for different levels of granularity. In these three cases ten attributes and the following class representation were used:

$$\begin{tikzcd}
\begin{array}{c} Class_1 \\ at^1_{1,1}\ \ at^2_{1,2} \\ m^1_{1,1}\ \ m^2_{1,2} \end{array} \arrow[d] & & \begin{array}{c} Class_3 \\ at^5_{3,1}\ \ at^6_{3,2} \\ m^5_{3,1}\ \ m^6_{3,2} \end{array} \arrow[dl] \arrow[dr] & \\
\begin{array}{c} Class_2 \\ at^3_{2,1}\ \ at^4_{2,2} \\ m^3_{2,1}\ \ m^4_{2,2} \end{array} \arrow[urr] & \begin{array}{c} Class_4 \\ at^7_{4,1}\ \ at^8_{4,2} \\ m^7_{4,1}\ \ m^8_{4,2} \end{array} \arrow[l] & & \begin{array}{c} Class_5 \\ at^9_{5,1}\ \ at^{10}_{5,2} \\ m^9_{5,1}\ \ m^{10}_{5,2} \end{array}
\end{tikzcd}$$

Here, two complex attributes are presented: $at^3_{2,1}$ and $at^7_{4,1}$. The following calls are made using these complex attributes:

- Method $m^3_{2,1}$ calls twice $at^3_{2,1}.m^5_{3,1}$
- Method $m^4_{2,2}$ calls once $at^3_{2,1}.m^6_{3,2}$
- Method $m^7_{4,1}$ calls once $at^7_{4,1}.m^3_{2,1}$
- Method $m^8_{4,2}$ calls twice $at^7_{4,1}.m^4_{2,2}$

For each example the TMUM shows the values obtained after multiplication of TMUM and frequencies of accessing of each transaction.

6.2 Experiment Data and Results

6.2.1 Simple attributes and simple methods

.

For this case several sub-cases were developed, which allow us to check the obtained fragmentation with the theoretical analysis previous to program run.

- Example 1 - Clear distribution of frequencies of accessing the methods by transaction.

  The following TMUM was used:

$$\begin{array}{c|cccccccccc}
 & m^1_{1,1} & m^2_{1,2} & m^3_{1,3} & m^4_{2,1} & m^5_{2,2} & m^6_{2,3} & m^7_{3,1} & m^8_{3,2} & m^9_{3,3} & m^{10}_{3,4} \\
tr_1 & 25 & 25 & 25 & 0 & 0 & 25 & 0 & 0 & 0 & 0 \\
tr_2 & 30 & 30 & 30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
tr_3 & 0 & 0 & 0 & 80 & 80 & 80 & 0 & 0 & 0 & 0 \\
tr_4 & 0 & 0 & 0 & 70 & 70 & 70 & 0 & 0 & 0 & 0 \\
tr_5 & 5 & 0 & 0 & 0 & 0 & 0 & 5 & 5 & 5 & 5 \\
tr_6 & 0 & 10 & 0 & 10 & 0 & 0 & 10 & 10 & 10 & 10
\end{array}$$

  This matrix shows a clear clustering of higher frequencies and a separation of lower frequencies. This facilitates the prediction of the fragmentation scheme.

  The used MAUM reflects the situation when the methods use the attributes only from their own class:

$$\begin{array}{ccccccccc}
 & at^1_{1,1} & at^2_{1,2} & at^3_{1,3} & at^4_{2,1} & at^5_{2,2} & at^6_{3,1} & at^7_{3,2} & at^8_{3,3} \\
m^1_{1,1} & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^2_{1,2} & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^3_{1,3} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
m^4_{2,1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
m^5_{2,2} & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
m^6_{2,3} & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 \\
m^7_{3,1} & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 \\
m^8_{3,2} & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \\
m^9_{3,3} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\
m^{10}_{3,4} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2
\end{array}$$

The values in MAUM are zero, one, or two to reflect no access, only read (retrieve), or read and write (update) of the attributes. The values can be changed in case it is necessary.

After multiplying the above matrices the resulted TAUM has the following values:

$$\begin{array}{ccccccccc}
 & at^1_{1,1} & at^2_{1,2} & at^3_{1,3} & at^4_{2,1} & at^5_{2,2} & at^6_{3,1} & at^7_{3,2} & at^8_{3,3} \\
tr_1 & 50 & 50 & 25 & 50 & 25 & 0 & 0 & 0 \\
tr_2 & 60 & 60 & 30 & 0 & 0 & 0 & 0 & 0 \\
tr_3 & 0 & 0 & 0 & 240 & 240 & 10 & 0 & 0 \\
tr_4 & 0 & 8 & 0 & 210 & 210 & 0 & 0 & 0 \\
tr_5 & 10 & 0 & 0 & 0 & 0 & 15 & 25 & 20 \\
tr_6 & 0 & 20 & 0 & 10 & 0 & 30 & 50 & 40
\end{array}$$

In this TAUM matrix we can see that dominant values are presented in such a way that they should give the following grouping of the attributes:

- $at^1_{1,1}$, $at^2_{1,2}$ and $at^3_{1,3}$
- $at^4_{2,1}$ and $at^5_{2,2}$.
- $at^6_{3,1}$, $at^7_{3,2}$ and $at^8_{3,3}$.

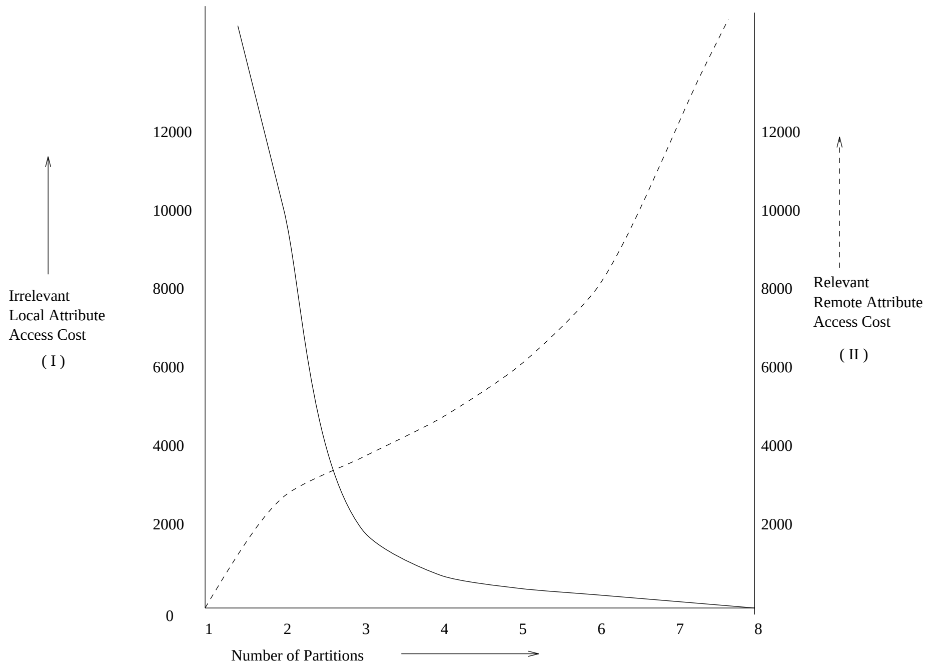


Figure 6.1. Behavior of the components of PEOO for an example

Indeed, the result of the program, using the criterion of minimum, gave this fragmentation as the optimal one with the minimum cost equal to 5307.

The local, remote, and total costs for the optimistic approach with the minimum values and the corresponding partitioning schemes are given below. The results for minimum cost of each component and total cost are plotted in fig. 6.1 and 6.2.

$NFrag$	$LocCost$	$RemCost$	$PEOO$	$Partition$
1	165028	0	165028	(12345678)
2	10408	2862	13270	(123678)(45)
3	1966	3341	5307	(123)(45)(678)
4	863	4950	5813	(12)(3)(45)678)
5	676	6075	6751	(12)(3)(45)(6)(78)
6	613	8075	8688	(12)(3)(45)(6)(7)(8)
7	363	13800	14163	(1)(2)(3)(45)(6)(7)(8)
8	0	116175	116175	(1)(2)(3)(4)(5)(6)(7)(8)

The behavior of PEOO corresponds to the previous supposition: the minimum value of local cost is equal to zero for one fragment and maximum for eight

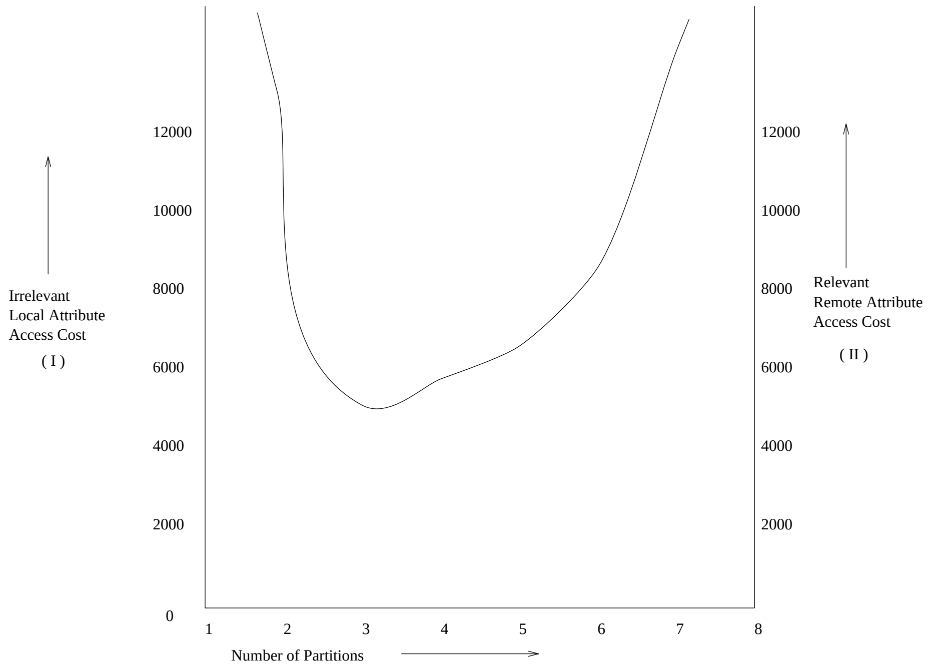


Figure 6.2. Behavior of PEOO for an example

fragments; the remote cost in these extremes has opposite values of maximum and minimum (equal to zero) respectively.

The additional experiment was run when the methods access attributes of their own class and their superclasses for the same TMUM. The values of MAUM were as follows:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{1,3}$	$at^4_{2,1}$	$at^5_{2,2}$	$at^6_{3,1}$	$at^7_{3,2}$	$at^8_{3,3}$
$m^1_{1,1}$	2	0	0	0	0	0	0	0
$m^2_{1,2}$	0	2	0	0	0	0	0	0
$m^3_{1,3}$	0	0	1	0	0	0	0	0
$m^4_{2,1}$	1	0	1	1	0	0	0	0
$m^5_{2,2}$	0	2	0	0	2	0	0	0
$m^6_{2,3}$	0	0	1	2	1	0	0	0
$m^7_{3,1}$	1	1	0	0	0	2	1	0
$m^8_{3,2}$	0	1	0	0	0	1	2	0
$m^9_{3,3}$	0	0	2	0	0	0	1	2
$m^{10}_{3,4}$	2	0	0	0	0	0	1	2

The obtained values for TAUM are presented as follows:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{1,3}$	$at^4_{2,1}$	$at^5_{2,2}$	$at^6_{3,1}$	$at^7_{3,2}$	$at^8_{3,3}$
tr_1	50	50	50	50	25	0	0	0
tr_2	60	60	30	0	0	0	0	0
tr_3	80	160	160	240	240	0	0	0
tr_4	70	140	140	210	210	0	0	0
tr_5	25	10	10	0	0	15	25	20
tr_6	40	40	30	10	0	30	50	40

In this example the optimal configuration was: (1678)(2345) with the minimum total cost equal to 35068.

- Example 2 - Influence of dominant value of transaction frequency.

  The partitioning scheme is sensitive to the presence of the significant values of the frequency of accessing method or attribute. For example, if we modify in the first example only one value for $m^6_{2,3}$ and $at^4_{2,1}$ in MMUM such as

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{1,3}$	$at^4_{2,1}$	$at^5_{2,2}$	$at^6_{3,1}$	$at^7_{3,2}$	$at^8_{3,3}$
$m^1_{1,1}$	2	0	0	0	0	0	0	0
$m^2_{1,2}$	0	2	0	0	0	0	0	0
$m^3_{1,3}$	0	0	1	0	0	0	0	0
$m^4_{2,1}$	0	0	0	1	0	0	0	0
$m^5_{2,2}$	0	0	0	0	2	0	0	0
$m^6_{2,3}$	0	0	0	0	1	0	0	0
$m^7_{3,1}$	0	0	0	0	0	2	1	0
$m^8_{3,2}$	0	0	0	0	0	1	2	0
$m^9_{3,3}$	0	0	0	0	0	0	1	2
$m^{10}_{3,4}$	0	0	0	0	0	0	1	2

leaving the matrix TMUM without changes, we will obtain TAUM with the corresponding values:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{1,3}$	$at^4_{2,1}$	$at^5_{2,2}$	$at^6_{3,1}$	$at^7_{3,2}$	$at^8_{3,3}$
tr_1	50	50	25	0	25	0	0	0
tr_2	60	60	30	0	0	0	0	0
tr_3	0	0	0	80	240	0	0	0
tr_4	0	0	0	0	210	0	0	0
tr_5	10	0	0	0	0	5	25	20
tr_6	0	20	0	10	0	30	50	40

This example gives the best fragmentation (123)(4)(5)(678), clearly putting the attribute $at^5_{2,2}$ in a different fragment than attributes $at^4_{2,1}$. This result was expected because the high frequency of accessing the attribute $at^5_{2,2}$ will increase the local cost (the part that corresponds to the accessing of relevant attributes with different frequencies) if these three attributes were together. This value changes from 1603 (with the total cost equal to 13794) for fragmentation (12)(3)(45)(678) to 24566 (with the total cost equal to 25094) for the fragmentation (123)(45)(678).

- Example 3 - Significant values of accessing the methods by transaction distributed more uniformly along the TMUM matrix.

  The other presented case uses values of frequencies of accessing methods by transaction distributed along the TMUM matrix without a clear clustering of significant and non-significant values as was done in the first example.

	$m^1_{1,1}$	$m^2_{1,2}$	$m^3_{1,3}$	$m^4_{2,1}$	$m^5_{2,2}$	$m^6_{2,3}$	$m^7_{3,1}$	$m^8_{3,2}$	$m^9_{3,3}$	$m^{10}_{3,4}$
tr_1	50	0	0	0	50	0	0	50	0	0
tr_2	0	30	30	0	0	0	30	0	0	0
tr_3	70	0	0	0	70	0	0	70	0	0
tr_4	0	50	50	0	0	0	50	0	0	0
tr_5	40	0	0	0	40	0	0	40	0	0
tr_6	0	0	0	10	0	10	0	0	10	10

Moreover, the MAUM was changed making one of the methods accessing in a dominant manner only one of the attributes. These changes were done to first analyze theoretically the expected result and then prove it against the program result.

The MAUM is the following:

$$\begin{array}{c|cccccccc}
 & at^1_{1,1} & at^2_{1,2} & at^3_{1,3} & at^4_{2,1} & at^5_{2,2} & at^6_{3,1} & at^7_{3,2} & at^8_{3,3} \\
m^1_{1,1} & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^2_{1,2} & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^3_{1,3} & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
m^4_{2,1} & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
m^5_{2,2} & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
m^6_{2,3} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
m^7_{3,1} & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
m^8_{3,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
m^9_{3,3} & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
m^{10}_{3,4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}$$

Analyzing these values, we can assume that the following methods have dominant access to the attributes:

method	*attribute*
$m^1_{1,1}$	$at^1_{1,1}$
$m^2_{1,2}$	$at^2_{1,2}$
$m^3_{1,3}$	$at^3_{1,3}$
$m^4_{2,1}$	$at^4_{2,1}$
$m^5_{2,2}$	$at^5_{2,2}$
$m^7_{3,1}$	$at^7_{3,2}$
$m^8_{3,2}$	$at^8_{3,3}$
$m^9_{3,3}$	$at^6_{3,1}$

The methods $m^6_{2,3}$ and $m^{10}_{3,4}$ do not have significant accesses to any of the attributes.

Theoretically, based on the values presented in TMUM and MAUM we can suppose:

– Transactions tr_1, tr_3 and tr_5 have dominant accesses to the methods $m^1_{1,1}$, $m^5_{2,2}$ and $m^8_{3,2}$. Considering the dominant accesses that these methods have, one of the fragments should include attributes $at^1_{1,1}$, $at^5_{2,2}$ and $at^8_{3,3}$.

- Transactions tr_2 and tr_4 access the methods $m^2_{1,2}$, $m^3_{1,3}$ and $m^7_{3,2}$ with higher frequencies. Based on this information and information about the attributes accessed by these methods we can expect that attributes $at^2_{1,2}$, $at^3_{1,3}$ and $at^7_{3,2}$ will be together in one fragment.
- Transaction tr_6 calls methods $m^4_{2,1}$ and $m^9_{3,3}$ putting the attributes $at^4_{2,1}$ and $at^6_{3,1}$ in one fragment. This transaction also accessed other methods. However, these other methods do not access any of the attributes in a dominant way.

After this analysis, the expected optimal partitioning is: (158)(237)(46)

Indeed, this result was obtained after running the program and the minimum PEOO values is equal to 201. The calculated values of TAUM were as follows:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{1,3}$	$at^4_{2,1}$	$at^5_{2,2}$	$at^6_{3,1}$	$at^7_{3,2}$	$at^8_{3,3}$
tr_1	100	0	0	0	100	0	0	100
tr_2	0	60	60	0	0	0	60	0
tr_3	140	0	0	0	140	0	0	140
tr_4	0	100	100	0	0	0	100	0
tr_5	80	0	0	0	80	8	0	80
tr_6	0	0	0	20	0	20	0	10

- Example 4 - Distribution of frequencies of accessing the methods by transactions with predictable distribution of two attributes per fragment.

  In this case the used TMUM and MMUM have the following values:

	$m^1_{1,1}$	$m^2_{1,2}$	$m^3_{1,3}$	$m^4_{2,1}$	$m^5_{2,2}$	$m^6_{2,3}$	$m^7_{3,1}$	$m^8_{3,2}$	$m^9_{3,3}$	$m^{10}_{3,4}$
tr_1	15	15	0	0	0	0	0	0	0	0
tr_2	25	25	0	0	0	0	0	0	0	0
tr_3	0	0	60	60	0	0	0	0	0	0
tr_4	0	0	40	40	0	0	0	0	0	0
tr_5	0	0	0	0	50	0	50	0	0	0
tr_6	0	0	0	0	0	0	0	45	0	45

$$
\begin{array}{c|cccccccc}
 & at^1_{1,1} & at^2_{1,2} & at^3_{1,3} & at^4_{2,1} & at^5_{2,2} & at^6_{3,1} & at^7_{3,2} & at^8_{3,3} \\
m^1_{1,1} & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^2_{1,2} & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^3_{1,3} & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
m^4_{2,1} & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
m^5_{2,2} & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
m^6_{2,3} & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
m^7_{3,1} & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 \\
m^8_{3,2} & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \\
m^9_{3,3} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
m^{10}_{3,4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2
\end{array}
$$

The calculated values of TAUM shows the clustering of dominant values in groups of two. They are presented as follows:

$$
\begin{array}{c|cccccccc}
 & at^1_{1,1} & at^2_{1,2} & at^3_{1,3} & at^4_{2,1} & at^5_{2,2} & at^6_{3,1} & at^7_{3,2} & at^8_{3,3} \\
tr_1 & 30 & 45 & 0 & 0 & 0 & 0 & 0 & 0 \\
tr_2 & 50 & 75 & 0 & 0 & 0 & 0 & 0 & 0 \\
tr_3 & 0 & 0 & 120 & 120 & 0 & 0 & 0 & 0 \\
tr_4 & 0 & 0 & 80 & 80 & 0 & 0 & 0 & 0 \\
tr_5 & 0 & 0 & 0 & 0 & 100 & 100 & 50 & 0 \\
tr_6 & 0 & 0 & 0 & 0 & 0 & 45 & 90 & 90
\end{array}
$$

The optimal configuration is (12)(34)(56)(78) with the minimum value of PEOO equal to 4951.

- Example 5 - Generic cases: We consider two generic cases when each fragment contains one attribute and when all attributes belong to the same fragment. For the first case, the calculated TAUM matrix that serves as a base for PEOO should have one dominant value for each attribute. In the second case, these values should be very similar for all transaction and all attributes. For both generic cases the MAUM is the same as presented in the previous case.

  1. The optimal fragmentation scheme giving one attribute per fragment

The values of TMUM, MMUM, and calculated TAUM are the following:

	$m_{1,1}^1$	$m_{1,2}^2$	$m_{1,3}^3$	$m_{2,1}^4$	$m_{2,2}^5$	$m_{2,3}^6$	$m_{3,1}^7$	$m_{3,2}^8$	$m_{3,3}^9$	$m_{3,4}^{10}$
tr_1	100	100	0	0	0	0	0	0	0	0
tr_2	0	0	10	0	0	0	0	0	0	0
tr_3	0	0	0	300	0	0	0	0	0	0
tr_4	0	0	0	0	50	50	0	0	0	0
tr_5	0	0	0	0	0	0	70	70	70	0
tr_6	0	0	0	0	0	0	0	0	0	80

	$at_{1,1}^1$	$at_{1,2}^2$	$at_{1,3}^3$	$at_{2,1}^4$	$at_{2,2}^5$	$at_{3,1}^6$	$at_{3,2}^7$	$at_{3,3}^8$
$m_{1,1}^1$	2	0	0	0	0	0	0	0
$m_{1,2}^2$	1	0	1	0	0	0	0	0
$m_{1,3}^3$	0	2	0	0	0	0	0	0
$m_{2,1}^4$	0	0	0	2	0	0	0	0
$m_{2,2}^5$	0	0	0	0	1	0	0	0
$m_{2,3}^6$	0	0	0	0	1	0	0	0
$m_{3,1}^7$	0	0	0	0	0	1	0	1
$m_{3,2}^8$	0	0	0	0	0	2	0	0
$m_{3,3}^9$	0	0	0	0	0	0	1	0
$m_{3,4}^{10}$	0	0	0	0	0	0	0	2

	$at_{1,1}^1$	$at_{1,2}^2$	$at_{1,3}^3$	$at_{2,1}^4$	$at_{2,2}^5$	$at_{3,1}^6$	$at_{3,2}^7$	$at_{3,3}^8$
tr_1	300	0	100	0	0	0	0	0
tr_2	0	20	0	0	0	0	0	0
tr_3	0	0	0	600	0	0	0	0
tr_4	0	0	0	0	100	0	0	0
tr_5	0	0	0	0	0	210	70	70
tr_6	0	0	0	0	0	0	0	160

Here, the optimal fragmentation (1)(2)(3)(4)(5)(6)(7)(8) give the minimum value of 19800 with the behavior of PEOO function described before.

2. The optimal fragmentation putting all attributes in one fragment

	$m^1_{1,1}$	$m^2_{1,2}$	$m^3_{1,3}$	$m^4_{2,1}$	$m^5_{2,2}$	$m^6_{2,3}$	$m^7_{3,1}$	$m^8_{3,2}$	$m^9_{3,3}$	$m^{10}_{3,4}$
tr_1	10	10	10	10	0	10	10	0	10	0
tr_2	20	20	20	0	20	0	20	20	0	20
tr_3	0	30	0	30	0	30	30	30	0	30
tr_4	50	50	50	0	50	50	50	50	0	50
tr_5	70	70	70	0	70	0	70	70	70	70
tr_6	0	80	80	80	0	80	0	80	80	80

The values of MAUM are:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{1,3}$	$at^4_{2,1}$	$at^5_{2,2}$	$at^6_{3,1}$	$at^7_{3,2}$	$at^8_{3,3}$
$m^1_{1,1}$	2	0	0	0	0	0	0	0
$m^2_{1,2}$	0	2	0	0	0	0	0	0
$m^3_{1,3}$	0	0	1	0	0	0	0	0
$m^4_{2,1}$	0	0	0	2	0	0	0	0
$m^5_{2,2}$	0	0	0	0	1	0	0	0
$m^6_{2,3}$	0	0	0	0	1	0	0	0
$m^7_{3,1}$	0	0	0	0	0	0	2	0
$m^8_{3,2}$	0	0	0	0	0	2	0	0
$m^9_{3,3}$	0	0	0	0	0	0	1	0
$m^{10}_{3,4}$	0	0	0	0	0	0	0	2

The calculated values of TAUM are:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{1,3}$	$at^4_{2,1}$	$at^5_{2,2}$	$at^6_{3,1}$	$at^7_{3,2}$	$at^8_{3,3}$
tr_1	20	20	10	20	10	0	30	0
tr_2	40	40	20	0	20	40	40	40
tr_3	0	60	0	60	30	60	60	60
tr_4	100	100	50	0	100	100	100	100
tr_5	140	140	70	0	70	140	210	140
tr_6	0	160	80	160	80	160	80	160

The optimal configuration in this case is (12345678) with the minimum values of PEOO equal to 70960.

6.2.2 Complex attributes and complex methods

.

The examples presented in this part will show that our hypothesis about eliminating the complex attribute from the analysis can be used in a fragmentation scheme, allowing us to decrease the number of attributes in the execution of the program, thus decreasing the execution time, in our case, of exhaustive search. In the real case when many complex attributes can be used, the improvement will be greater. Moreover, if some heuristics are developed in the future, this fact can also have influence on the execution time.

We present three cases:

1. The complex attribute $at^3_{2,1}$ is treated as a usual attribute. The MAUM reflects the frequencies of accessing this attribute by methods of the $Class_2$. Calls to the methods of the class that this attribute is pointing to are also reflected in MMUM as a case of nested method calls.

2. The complex attribute is presented in MAUM, however, the frequencies of accessing this attribute by methods $m^4_{2,1}$ and $m^5_{2,2}$ are eliminated (putting values of zero). This allows us to consider as feasible the possibility of ignoring this attribute. The idea behind this is that because this attribute does not represent any physical storage, the access to it can be ignored.

3. The complex attribute is eliminated from the analysis, which gives only seven attributes to consider instead of eight. All the possible combinations with the number of fragments varying from one to seven are analyzed.

In all the three cases the following TMUM and MMUM matrices are used:

	$m^1_{1,1}$	$m^2_{1,2}$	$m^3_{1,3}$	$m^4_{2,1}$	$m^5_{2,2}$	$m^6_{2,3}$	$m^7_{3,1}$	$m^8_{3,2}$	$m^9_{4,1}$	$m^{10}_{4,2}$	$m^{11}_{4,3}$
tr_1	0	25	0	25	0	0	0	0	25	25	0
tr_2	50	0	0	50	0	50	0	0	0	50	0
tr_3	0	15	15	0	0	0	15	15	0	0	15
tr_4	0	0	0	0	35	35	35	0	0	35	35
tr_5	25	25	0	0	0	0	25	0	0	25	0

$$\begin{array}{c|ccccccccccc}
 & m^1_{1,1} & m^2_{1,2} & m^3_{1,3} & m^4_{2,1} & m^5_{2,2} & m^6_{2,3} & m^7_{3,1} & m^8_{3,2} & m^9_{4,1} & m^{10}_{4,2} & m^{11}_{4,3} \\
m^1_{1,1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^2_{1,2} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^3_{1,3} & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^4_{2,1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
m^5_{2,2} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
m^6_{2,3} & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
m^7_{3,1} & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
m^8_{3,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
m^9_{4,1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
m^{10}_{4,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
m^{11}_{4,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{array}$$

It is important to emphasize the existence of the values in MMUM in the positions (4,9), (4,11) and (5,10) that reflect the nested calls between the following methods:

1. $m^4_{2.1}$ and $m^9_{4,1}$

2. $m^4_{2,1}$ and $m^{11}_{4,3}$

3. $m^5_{2,2}$ and $m^{10}_{4,2}$

There are also presented other nested method calls, such as: $m^7_{3,1}$ and $m^2_{1,2}$, $m^9_{4,1}$ and $m^7_{3,1}$, and so forth.

The MAUM is different in all cases and in the first one is presented as:

$$\begin{array}{c|cccccccc}
 & at^1_{1,1} & at^2_{1,2} & at^3_{2,1} & at^4_{2,2} & at^5_{3,1} & at^6_{3,2} & at^7_{4,1} & at^8_{4,2} \\
m^1_{1,1} & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^2_{1,2} & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^3_{1,3} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^4_{2,1} & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\
m^5_{2,2} & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\
m^6_{2,3} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
m^7_{3,1} & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
m^8_{3,2} & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
m^9_{4,1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
m^{10}_{4,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
m^{11}_{4,3} & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0
\end{array}$$

Notice the presence of the frequencies in the positions (4,3) and (5,1) that reflect the use of attribute $at^3_{2,1}$ in methods $m^4_{2,1}$ and $m^5_{2,2}$.

In the second case this matrix has the following values:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{2,1}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$
$m^1_{1,1}$	2	1	0	0	0	0	0	0
$m^2_{1,2}$	0	2	0	0	0	0	0	0
$m^3_{1,3}$	1	1	0	0	0	0	0	0
$m^4_{2,1}$	0	0	0	1	0	0	0	0
$m^5_{2,2}$	0	0	0	2	0	0	0	0
$m^6_{2,3}$	0	0	0	1	0	0	0	0
$m^7_{3,1}$	0	0	0	0	2	0	0	0
$m^8_{3,2}$	0	0	0	0	1	1	0	0
$m^9_{4,1}$	0	0	0	0	0	0	1	0
$m^{10}_{4,2}$	0	0	0	0	0	0	0	2
$m^{11}_{4,3}$	0	0	0	0	0	0	2	0

Here, two changes were done in the positions (4,3) and (5,3) resulting column with attribute $at^3_{2,1}$ with only values of zero.

For the third case, after the elimination of the attribute $at^3_{2,1}$, the MAUM uses the following values:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$
$m^1_{1,1}$	2	1	0	0	0	0	0
$m^2_{1,2}$	0	2	0	0	0	0	0
$m^3_{1,3}$	1	1	0	0	0	0	0
$m^4_{2,1}$	0	0	1	0	0	0	0
$m^5_{2,2}$	0	0	2	0	0	0	0
$m^6_{2,3}$	0	0	1	0	0	0	0
$m^7_{3,1}$	0	0	0	2	0	0	0
$m^8_{3,2}$	0	0	0	1	1	0	0
$m^9_{4,1}$	0	0	0	0	0	1	0
$m^{10}_{4,2}$	0	0	0	0	0	0	2
$m^{11}_{4,3}$	0	0	0	0	0	2	0

It is interesting to analyze the obtained TAUM values after multiplication (TMUM, MMUM and MAUM) in these three cases:

For the first case this matrix has the following values:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{2,1}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$
tr_1	50	75	50	25	50	0	100	50
tr_2	100	50	200	150	0	0	150	100
tr_3	75	135	30	15	45	15	30	30
tr_4	0	70	175	175	70	0	70	210
tr_5	100	150	50	25	50	0	0	50

In the second case with values of zero in the third column of MAUM, the TAUM values are exactly the same as in the previous example, except for the third column in which these values are also equal to zero:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{2,1}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$
tr_1	50	75	0	25	50	0	100	50
tr_2	100	50	0	150	0	0	150	100
tr_3	75	135	0	15	45	15	30	30
tr_4	0	70	0	175	70	0	70	210
tr_5	100	150	0	25	50	0	0	50

In the third case, with the elimination of the complex attributes, the calculated MAUM has the same values in all positions as in both matrices presented above, except the third column from the previous cases. Here this column does not exist:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$
tr_1	50	75	25	50	0	100	50
tr_2	100	50	150	0	0	150	100
tr_3	75	135	15	45	15	30	30
tr_4	0	70	175	70	0	70	210
tr_5	100	150	25	50	0	0	50

The obtained optimal fragmentation schemes and PEOO values are as follows:

Case	*Fragmentation*	*MinPEOO*
1	(12) (3478) (5) (6)	70314
2	(12) (36) (478) (5)	62998
	(12) (3) (478) (5) (6)	62998
3	(12) (478) (5) (6)	62998

6.2.3 Complex attributes and simple methods

To further confirm our hypothesis of eliminating complex attributes, we prepare one more experiment, where complex methods are not presented, except the ones that reflect calls through the complex attribute. For the exact same values of TMUM, presented in the previous case of complex attributes and complex methods, the following MMUM was developed:

$$\begin{array}{c|ccccccccccc}
 & m^1_{1,1} & m^2_{1,2} & m^3_{1,3} & m^4_{2,1} & m^5_{2,2} & m^6_{2,3} & m^7_{3,1} & m^8_{3,2} & m^9_{4,1} & m^{10}_{4,2} & m^{11}_{4,3} \\
m^1_{1,1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^2_{1,2} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^3_{1,3} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
m^4_{2,1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
m^5_{2,2} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
m^6_{2,3} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
m^7_{3,1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
m^8_{3,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
m^9_{4,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
m^{10}_{4,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
m^{11}_{4,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}$$

This matrix is presented only to reflect the access to the attributes of the $Class_4$ through the complex attribute of $Class_2$. We can recall from chapter 4 that this case of complex attributes and simple methods, after applying our hypothesis, converts to the case of simple attributes and complex methods.

The TMUM presented in the example before along with the above values of MMUM were used. The values of MAUM were different for both cases and are presented below:

1. All 8 attributes are presented and MAUM has the following values:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{2,1}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$
$m^1_{1,1}$	2	1	0	0	0	0	0	0
$m^2_{1,2}$	0	2	0	0	0	0	0	0
$m^3_{1,3}$	1	1	0	0	0	0	0	0
$m^4_{2,1}$	0	0	2	1	0	0	0	0
$m^5_{2,2}$	0	0	1	2	0	0	0	0
$m^6_{2,3}$	0	0	0	1	0	0	0	0
$m^7_{3,1}$	0	0	0	0	2	0	0	0
$m^8_{3,2}$	0	0	0	0	1	1	0	0
$m^9_{4,1}$	0	0	0	0	0	0	1	0
$m^{10}_{4,2}$	0	0	0	0	0	0	0	2
$m^{11}_{4,3}$	0	0	0	0	0	0	2	0

2. Seven attributes are used (complex attribute is eliminated) with the following values:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$
$m^1_{1,1}$	2	1	0	0	0	0	0
$m^2_{1,2}$	0	2	0	0	0	0	0
$m^3_{1,3}$	1	1	0	0	0	0	0
$m^4_{2,1}$	0	0	1	0	0	0	0
$m^5_{2,2}$	0	0	2	0	0	0	0
$m^6_{2,3}$	0	0	1	0	0	0	0
$m^7_{3,1}$	0	0	0	2	0	0	0
$m^8_{3,2}$	0	0	0	1	1	0	0
$m^9_{4,1}$	0	0	0	0	0	1	0
$m^{10}_{4,2}$	0	0	0	0	0	0	2
$m^{11}_{4,3}$	0	0	0	0	0	2	0

In both cases the calculated TAUM differs only in the position of the third column, where in the second case this column is eliminated.

These values for the first example are as follows:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{2,1}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$
tr_1	0	50	50	25	0	0	100	50
tr_2	100	50	100	100	0	0	150	100
tr_3	15	45	0	0	45	15	30	0
tr_4	0	0	35	105	70	0	70	140
tr_5	50	75	0	0	50	0	0	50

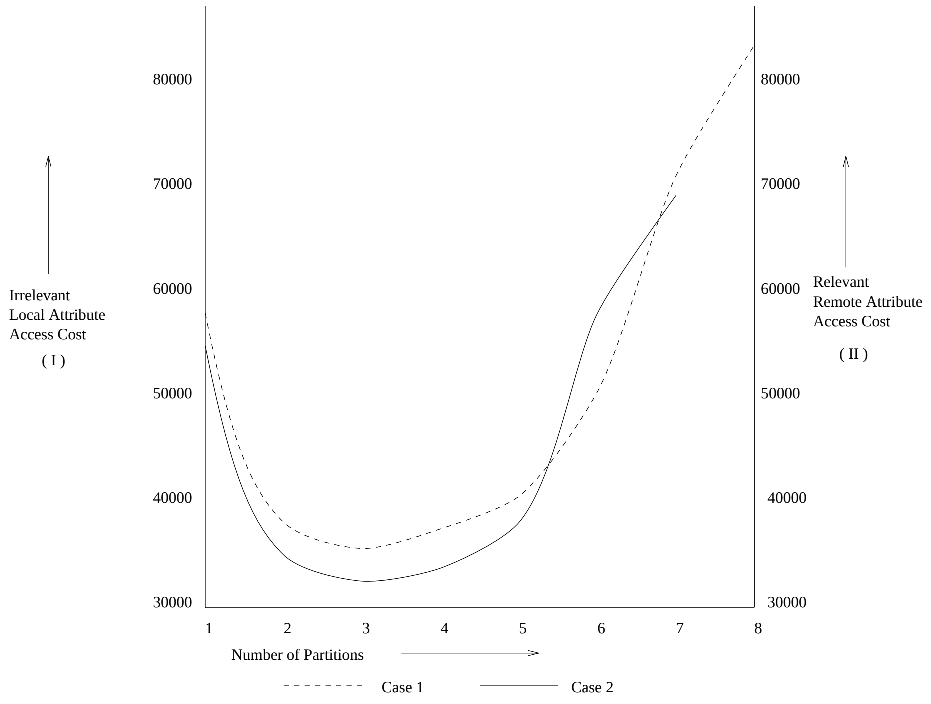


Figure 6.3. Behavior of PEOO for both examples

And, if we eliminate the third column, the remaining values represent the TAUM for the second example.

The minimum total cost with corresponding fragmentation are presented below:

Case	*Fragmentation*	*MinPEOO*
1	(125) (3478) (6)	36349
2	(125) (478) (6)	32412

The behavior of PEOO for both examples are plotted in fig 6.3.

6.2.4 Different levels of granularity

In this section we will present the fragmentation schemes obtained after applying our PEOO to the different levels of hierarchical structure of classes, which are presented at the beginning of this chapter. Three experiments were conducted for the following granularity levels:

- Granularity level equal to two for the both root classes. In this case all ten attributes were considered to be fragmented.

- Granularity level equal to one for both root classes. Here the attributes of $Class_2$, $Class_4$ and $Class_5$ should be seen as indivisible entities.
- Granularity level equal to two for the root $Class_1$, and zero for the root $Class_3$. Here, all the attributes of the $Class_3$, $Class_4$ and $Class_5$ were considered as indivisible entities.

For the granularity level equal to two for both root classes the corresponding TMUM, MMAUM and MAUM were used:

	$m^1_{1,1}$	$m^2_{1,2}$	$m^3_{2,1}$	$m^4_{2,2}$	$m^5_{3,1}$	$m^6_{3,2}$	$m^7_{4,1}$	$m^8_{4,2}$	$m^9_{5,1}$	$m^{10}_{5,2}$
tr_1	15	0	15	0	0	15	15	0	0	15
tr_2	0	7	0	7	0	0	7	0	0	7
tr_3	12	0	0	12	12	0	0	12	0	12
tr_4	0	9	9	0	9	0	9	0	9	0
tr_5	0	0	0	10	0	10	10	10	10	0

	$m^1_{1,1}$	$m^2_{1,2}$	$m^3_{2,1}$	$m^4_{2,2}$	$m^5_{3,1}$	$m^6_{3,2}$	$m^7_{4,1}$	$m^8_{4,2}$	$m^9_{5,1}$	$m^{10}_{5,2}$
$m^1_{1,1}$	1	0	0	0	0	0	0	0	0	0
$m^2_{1,2}$	1	1	0	0	0	0	0	0	0	0
$m^3_{2,1}$	0	1	1	0	2	0	0	0	0	0
$m^4_{2,2}$	1	0	0	1	0	1	0	0	0	0
$m^5_{3,1}$	0	0	0	0	1	0	0	0	0	0
$m^6_{3,1}$	0	0	0	0	0	1	0	0	0	0
$m^7_{4,1}$	0	0	1	0	1	1	1	0	0	0
$m^8_{4,2}$	0	0	0	2	0	1	0	1	0	0
$m^9_{5,1}$	0	0	0	0	1	0	0	0	1	1
$m^{10}_{5,2}$	0	0	0	0	1	1	0	0	0	1

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{2,1}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$	$at^9_{5,1}$	$at^{10}_{5,2}$
$m^1_{1,1}$	1	0	0	0	0	0	0	0	0	0
$m^2_{1,2}$	1	2	0	0	0	0	0	0	0	0
$m^3_{2,1}$	0	0	0	2	0	0	0	0	0	0
$m^4_{2,2}$	0	1	0	1	0	0	0	0	0	0
$m^5_{3,1}$	0	0	0	0	2	1	0	0	0	0
$m^6_{3,2}$	0	0	0	0	0	1	0	0	0	0
$m^7_{4,1}$	0	0	0	0	1	0	0	1	0	0
$m^8_{4,2}$	0	0	0	0	0	1	0	2	0	0
$m^9_{5,1}$	0	0	0	0	1	1	0	0	1	2
$m^{10}_{5,2}$	0	0	0	0	0	0	0	0	2	1

Notice that the frequencies of accessing complex attributes are equal to zero for the reasons explained before (these attributes do not represent any access cost to physical storage).

The calculated values of TAUM were as follows:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{2,1}$	$at^4_{2,2}$	$at^5_{3,1}$	$at^6_{3,2}$	$at^7_{4,1}$	$at^8_{4,2}$	$at^9_{5,1}$	$at^{10}_{5,2}$
tr_1	30	30	0	60	135	105	0	15	30	15
tr_2	21	21	0	21	35	35	0	7	14	7
tr_3	24	36	0	36	48	72	0	24	24	12
tr_4	27	36	0	36	108	63	0	9	27	27
tr_5	10	30	0	30	30	60	0	20	30	30

After checking the 115975 possible combinations of ten attributes, we obtain the following optimal fragmentation schemes: (137)(2456)(8)(9)(10), (13)2456)(8)(9)(10), (17)(2456)(3)(8)(9)(10), and (1)(2456)(3)(7)(8)(9)(10) all with the same total minimum cost equal to 21152.

The same result ((1)(2456)(8)(9)(10)) was obtained after eliminating from MAUM both complex attributes and running the program with only eight attributes and 4140 cases.

For the second example when the granularity level was equal to one for both root classes the TMUM, MUMUM and MAUM should be recalculated to reflect the indivisibility of some group of attributes. In our case we will have:

	$m_{1,1}^1$	$m_{1,2}^2$	$m_{3,1}^5$	$m_{3,2}^6$	M_2	M_4	M_5
tr_1	15	0	0	15	15	15	15
tr_2	0	7	0	0	7	7	7
tr_3	12	0	12	0	12	12	12
tr_4	0	9	9	0	9	9	9
tr_5	0	0	0	10	10	20	10

Here, the values of the position corresponding to columns M_i represent the sum of the frequencies of accessing methods that belong to the specified group. For example, the value equal to twenty in the position (5,6) is the sum of the values ten and ten from the positions (5,7) and (5,8) respectively of the TMUM from the first example (the methods $m_{4,1}^7$ and $m_{4,2}^8$ belong to the group M_4).

In a similar way the values of MMUM were recalculated.

	$m_{1,1}^1$	$m_{1,2}^2$	$m_{3,1}^5$	$m_{3,2}^6$	M_2	M_4	M_5
$m_{1,1}^1$	1	0	0	0	0	0	0
$m_{1,2}^2$	1	1	0	0	0	0	0
$m_{3,1}^5$	0	0	1	0	0	0	0
$m_{3,2}^6$	0	0	0	1	0	0	0
M_2	1	1	2	1	1	0	0
M_4	0	0	1	2	3	1	0
M_5	0	0	2	1	0	0	2

It is important to notice that if there are no calls between the methods that belong to the same group, the value in the diagonal will be equal to one. In the other case, if these calls exist, the value in the diagonal represents the sum of the frequencies of the methods that belong to the same group and call each other. In our example, in the last position (7,7) the value is equal to two, because from the previous example we can see that the method $m_{5,1}^9$ calls the method $m_{5,2}^{10}$ and that both of these methods belong to the same group M_5. Other values are calculated according to the explanation given in chapter 4 for the case of simple attributes and simple methods. For example, the value of three in the position (6,5) represents the sum of the frequencies of the calls of the methods $m^7_{4,1}$ and $m^8_{4,2}$ to the methods $m^3_{2,1}$ and $m^4_{2,2}$.

The recalculated values of MAUM according to the existing group are the following:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^5_{3,1}$	$at^6_{3,2}$	AT_2	AT_4	AT_5
$m^1_{1,1}$	1	0	0	0	0	0	0
$m^2_{1,2}$	1	2	0	0	0	0	0
$m^5_{3,1}$	0	0	2	1	0	0	0
$m^6_{3,2}$	0	0	0	1	0	0	0
M_2	0	1	0	0	3	0	0
M_4	0	0	1	1	0	3	0
M_5	0	0	1	1	0	0	6

After matrix multiplication the values of TAUM are as follows:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^5_{3,1}$	$at^6_{3,2}$	AT_2	AT_4	AT_5
tr_1	35	90	195	195	180	45	180
tr_2	28	56	91	84	84	21	84
tr_3	36	72	180	156	144	36	144
tr_4	36	72	135	117	108	27	108
tr_5	20	90	160	170	210	60	120

The optimal fragmentation scheme is: (1)(23456910)(78) with the total minimum cost of 38885.

For the last examples with the granularity level equal to two for the root $Class_1$ and zero for the root $Class_3$ the corresponding matrices are used:

TMUM:

	$m^1_{1,1}$	$m^2_{1,2}$	$m^3_{2,1}$	$m^4_{2,2}$	M_3
tr_1	15	0	15	0	45
tr_2	0	7	0	7	14
tr_3	12	0	0	12	36
tr_4	0	9	9	0	27
tr_5	0	0	0	10	40

MMUM:

	$m^1_{1,1}$	$m^2_{1,2}$	$m^3_{2,1}$	$m^4_{2,2}$	M_3
$m^1_{1,1}$	1	0	0	0	0
$m^2_{1,2}$	1	1	0	0	0
$m^3_{2,1}$	0	1	1	0	2
$m^4_{2,2}$	1	0	0	1	1
M_3	0	0	1	2	13

MAUM:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{2,1}$	$at^4_{2,2}$	AT_3
$m^1_{1,1}$	1	0	0	0	0
$m^2_{1,2}$	1	2	0	0	0
$m^3_{2,1}$	0	0	0	2	0
$m^4_{2,2}$	0	1	0	1	0
M_3	0	0	0	0	17

The calculated values of TAUM are:

	$at^1_{1,1}$	$at^2_{1,2}$	$at^3_{2,1}$	$at^4_{2,2}$	AT_3
tr_1	30	120	0	210	10455
tr_2	21	49	0	63	3213
tr_3	24	84	0	156	8160
tr_4	27	90	0	126	6273
tr_5	10	90	0	170	9010

The optimal fragmentation scheme is: (13)(2)(4)(5678910) or (1)(2)(3)(4)(5678910) with the total minimum cost of 159984.

6.3 Interpretation of the Results

The experiments presented for simple attributes and simple methods allow the reader to see the expected results after program execution. Considering several cases, such as clustering of frequencies of accessing methods by transactions, more uniform distribution of the values, dominant frequency for one specific attribute, and so forth different fragmentation schemes are obtained. In each case the behavior of PEOO was analyzed and was found similar to the PE behavior presented by Chakravarthy et al. [4] and expected for PEOO.

The case of simple attributes and complex methods was not presented because its theoretical analysis should be done based on the calculated values of TAUM and not from the analysis of input matrices, TMUM and MAUM. This limits our theoretical analysis to the case of simple attributes and simple methods with varying values in TAUM such as those presented in the previous case.

The complex attribute and simple method, and complex attributes and complex methods can be analyzed as a case of simple attributes and complex methods, after eliminating the complex attribute from our consideration.

In the case of complex attributes and complex methods we consider three different examples as was described in the previous section.

In the first example the value of PEOO is higher because of the incorporated frequencies of accessing complex attribute by corresponding methods. However, this complex attributes in reality does not represent physical disk storage and it will be presented as a pointer in main memory when the database is open ([23, 2, 22]). Because our PEOO does not distinguish main memory attributes and all accesses are treated as accesses to the disk, even the access to the main memory attribute will be consider in the same way as access to the attributes presented in the disk, increasing the minimum value of PEOO. Therefore, the values for complex attributes should be equal to zero if we want to conserve the number of attributes considered in the fragmentation analysis, or it should be eliminated from this analysis. In the other case, when this attribute is treated the same way as simple attribute, the fragmentation scheme and its cost can be unnecessarily affected. According to our PEOO, the local cost reflects the access cost to disk storage in units; this cannot be applied in the same way to the complex attributes that are accessed from the main memory.

The second and third examples give the same values of PEOO. In addition, the complex attribute presented in the second case (attribute 3) is allocated with the attribute 6 because in the TAUM the values presented in the third and sixth columns are closer than others and give less penalty for the local cost if this attribute were allocated to another fragment.

For example, if this matrix has the following values:

$$\begin{array}{lrrrrrrrr}
 & at^1_{1,1} & at^2_{1,2} & at^3_{2,1} & at^4_{2,2} & at^5_{3,1} & at^6_{3,2} & at^7_{4,1} & at^8_{4,2} \\
tr_1 & 50 & 75 & 0 & 25 & 0 & 50 & 100 & 50 \\
tr_2 & 100 & 50 & 0 & 150 & 0 & 25 & 150 & 100 \\
tr_3 & 75 & 105 & 0 & 0 & 5 & 15 & 30 & 30 \\
tr_4 & 0 & 0 & 0 & 140 & 5 & 50 & 70 & 210 \\
tr_5 & 100 & 100 & 0 & 0 & 0 & 50 & 0 & 50
\end{array}$$

when the fifth column has the smallest frequencies, the minimum cost will be 49723 and the corresponding fragmentation to this cost is (12)(35)(478)(6). The same cost was given for the other fragmentation scheme, where the complex attribute is located in a separate fragment: (12)(3)(478)(5)(6).

For the the last example of the case of complex attributes and complex methods, when the attribute 3 is not presented for the fragmentation algorithm, after the partitioning scheme is given, we can put this complex attribute in the same fragment where attribute 6 is presented, Therefore, we obtain the same fragmentation scheme with or without the presence of the complex attribute in development of the fragmentation scheme. Moreover, the third example use less combinations for finding the optimal fragmentation, and, as was mentioned before, in the case of the presence of the several complex attributes, the decreasing number of considered attributes can improve the execution time.

Following logic, we can assume that the complex attribute in reality can be allocated in any fragment that it is needed. This is because it can be replicated without any additional consequences of its updating.

There is one more case of complex attributes and simple methods with two examples, which were developed to confirm our hypothesis. Also, these examples gave the expected result. The explanation of increasing total cost for the case of eight attributes in comparison to seven attributes along with the presence of the complex attribute in the second fragment is exactly the same as presented in other examples for the case of complex attributes and complex methods.

Therefore, our hypothesis of considering the case of complex attributes and simple methods, and complex attributes and complex methods as a case of simple attributes and complex methods, after the elimination of the complex attributes from fragmentation algorithms, was demonstrated to be true.

The last part of the presented examples shows the feasibility of using different levels of granularity. This allows the designer to not partition the "critical" classes that should not be fragmented.

CHAPTER 7
CONCLUSION AND POSSIBLE EXTENSION

7.1 Summary

The development of the Object-Oriented Partition Evaluator allows us to show the possibility of generalization and adoption of some concepts from the relational approach to the object-oriented approach. The concept of vertical partitioning, when extended to the object-oriented database systems as presented in this thesis, opens the possibility to "flatten" the complicated hierarchical structure and present it in the form of matrices, which are easy to manage for computational purposes. These matrices represent the relationship between transactions, methods and attributes. Moreover, the method-attribute usage matrix and the concepts of encapsulation and information hiding adopted from programming languages gives the designer the possibility to distinguish between retrieve and update operations without special modifications to the presented algorithms.

Additionally, the presented hypothesis of eliminating the complex attributes from the vertical fragmentation scheme reduces the complexity of the system presented for the partitioning. These attributes are eliminated only for calculation purposes and are put back again into the fragments where they are needed, after the fragmentation scheme is obtained. They can be allocated in any site because for them there is no cost of retrieving data from the disk. They can be replicated the same way as methods without any additional cost of future updating.

Moreover, different levels of granularity applied to the system allow the user to put some additional restrictions regarding the partitioning of the classes, limiting it to the desired hierarchy level.

The algorithm for vertical partitioning for complex attributes and complex methods represents the generalization of three other cases of simple attributes and simple methods, simple attributes and complex methods, complex attributes and simple methods. This means, one general program can be used for vertical partitioning and each of the corresponding options can be seen as a special case of complex attributes and complex methods.

The presented experiments confirm the expected behavior of PEOO and show feasibility of its use.

7.2 Future Work

Our approach assumes the presence of the statistics according to the frequencies of transactions, methods and attributes along with the information concerning the hierarchical structure of the system. In the case, in which other information is available, this PEOO can be adopted to the specific situations.

- *Fixed Partitioning Size.*

  If the number of desired fragments is known a priori, the program can be easily modified for this special situation.

- *Local and Remote Processing Cost*

  In our PEOO unit cost was assumed for calculating the local and remote cost. The local cost can depend on the access methods, but more importantly, the communication cost between sites can be different and can also be expressed as a ratio with respect to the local cost. In a real-life situation, working with the specified network configuration and knowing local access methods, additional analysis need to be done to calculate the value of PEOO to reflect actual costs.

- *Replication of objects.*

  Our PEOO does not consider the replication of objects. However, if the replication of data is recommended in some specific system, it can be applied and the corresponding data can be retried from the local node. In this case, our PEOO remains the same. However, the total cost will not include the additional cost of updating replicas and additional analysis should be done if this cost is considered important for system designers.

- *Allocation*

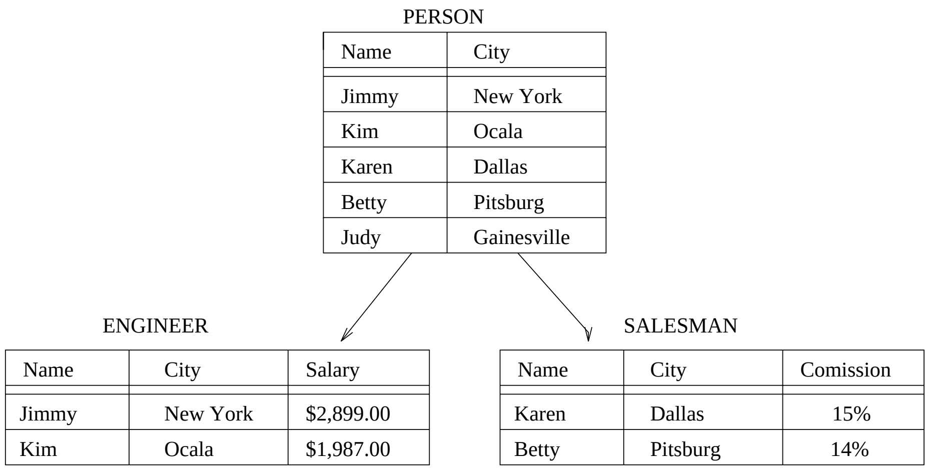
  To make our PEOO more universal, the extension should be done to analyze the allocation of the fragments to the specific sites.

- *Physical object allocation in hierarchy level*

  The partitioning presented in this thesis can be done whatever physical storage is chosen. According to Kim [20], there exists two ways of storing objects in class hierarchy:

  1. The object of subclasses are replicated in each of the levels of their superclasses (figure 7.1).
  2. The object physically exist only on the level of the class that they belong to (figure 7.2).

  Both of these approaches are used in different object-oriented systems, such as GALILEO, ADAPLEX (first approach) and ORION, GemStone (second approach).



PERSON

Name	City
Jimmy	New York
Kim	Ocala
Karen	Dallas
Betty	Pitsburg
Judy	Gainesville

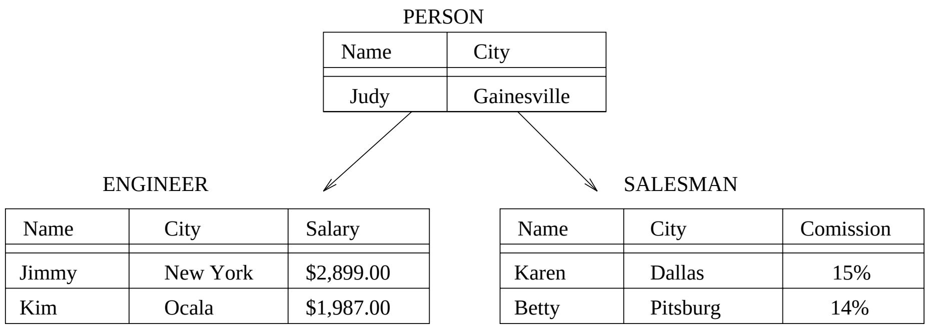
ENGINEER

Name	City	Salary
Jimmy	New York	$2,899.00
Kim	Ocala	$1,987.00

SALESMAN

Name	City	Comission
Karen	Dallas	15%
Betty	Pitsburg	14%

Figure 7.1. Replication of objects to their super-classes



PERSON

Name	City
Judy	Gainesville

ENGINEER

Name	City	Salary
Jimmy	New York	$2,899.00
Kim	Ocala	$1,987.00

SALESMAN

Name	City	Comission
Karen	Dallas	15%
Betty	Pitsburg	14%

Figure 7.2. Objects appear only once

It is obvious that the first approach allows fast access to the data from higher classes. The price for this is the replication of data and the problems related to it. In the second case, the problem of replication does not exist, but for accessing the data we should follow the path from the root to the corresponding class.

The proposed fragmentation scheme does not depend on physical storage that the system has implemented. However, because after fragmentation the option of replication of the objects presented on the level greater that granularity level can be chosen, the first option of the physical storage could have highly replicated objects and involves the known difficulties in managing replicas efficiently.

- *Object identity*

  The other problem related to the physical storage is the managing of the object identity.

  According to Kim [21] the logical, not physical OID should be used for the fragmentation of the objects.

  To manage OID, the OODB usually uses a global table of OIDs to ensure their uniqueness [30]. This global table can be used to show the fragments of the same object, having for each fragmented object a list of its new fragmented OID.

- *Presence of tuple, list and set*

  The use of these structures in OODB should not give any problem in expressing their frequencies of accessing in the form of TMUM, MMUM and MAUM. If this is the case, our PEOO can be applied without any changes to the system where these structures are presented.

These aspects were not analyzed in a detailed fashion and should be proposed as an extension to the presented work.

REFERENCES

[1] Atkinson, M., DeWitt, D., Maier, D., Bancilhon, F., Ditrich, K., and Zdonik, S. The Object-Oriented Database System Manifesto in F.Bancilhon, C.Delobel, and P.Kanellakis (eds.) *Building an Object-Oriented Database System: The Story of O_2*. Morgan Kaufman Publishers. San Mateo, CA, 1992.

[2] Carey, M.J., DeWitt, D.J., Franklin, M.J., Hall, N.E., McAuliffe, M.L., Naughton, J.F., Shuh, D.T., Solomon, C.K., Tan, C.K., Tsatalos, O. G., White, S.J., and Zwilling, M.J. Sharing Up Persistent Application. *Proceedings of the ACM SIGMOD, International Conference on Management of Data*, Vol.23, No.2, New York, June 1994.

[3] Ceri, S., and Pelagatti, G. *Distributed Databases: Principles and Systems.* McGraw-Hill, New York, 1984.

[4] Chakravarthy, S., Muthuraj, J., Varadarajan, R., and Navathe, S.B. An Objective Function for Vertically Partitioning Relations in Distributed Databases and its Analysis. *Distributed and Parallel Databases*, Vol.2, No.2, New York, April 1993.

[5] Chu Pai-Cheng. A Transaction Oriented Approach to Attribute Partitioning. *Information System*, Vol. 17, No.4, London, 1992.

[6] Chu, W., and Ieong, I.T. A Transaction-Based Approach to Vertical Partitioning for Relational Database Systems. *IEEE Transactions on Software Engineering*, Vol. 19, No. 8, New York, August 1993.

[7] Cornell, D., and Yu, P. A Vertical Partitioning Algorithm for Relational Databases. *Proceedings of the Third International Conference on Data Engineering*, Los Angeles, CA, February 1987.

[8] Date, C.J. *An Introducation to Database Systems.* Sixth edition, Addison-Wesley Publishing Company, New York, 1995.

[9] Elmasri, R., and Navathe, S.B. *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company, New York, 1994.

[10] Ezeife, C.I., and Barker, K. A Comprehensive Approach to Horizontal Class Fragmentation in a Distributed Object Based System. *Technical report.* Advanced Database Systems Laboratory, Department of Computer Science, University of Manitoba, Canada, October 1994.

[11] Golberg, A., and Robson,, D. *Smalltalk-80: The Language and its Implementation.* Addison-Wesley, Readings, MA, 1983.

[12] Gruber, O., and Amsaleg, L. Object Grouping in EOS in M.T.Ozsu, U.Dayal and P.Valduriez (eds.) *Distributed Object Management.* Morgan Kaufman Publishers, San Mateo, CA 1994.

[13] Hammer, M., and Niammir, B. A Heuristic Approach to Attributes Partitioning. *Proceedings ACM SIGMOD International Conference on Management of Data*, Boston, MA, 1979.

[14] Hoffer, J. A., and Severance, D. G. The Use of Cluster Analysis in Physical Database Design. *Proc. First International Conference on Very Large Data Bases*, Framingham, MA, September 1975.

[15] Jain, A., and Dubes, R. *Algorithms for Clustering Data.* Prentice Hall Advanced Reference Series, Englewood Cliffs, NJ, 1988.

[16] Kanellakis, P., Lecluse, Ch., and Richard, Ph. Introduction to Data Model in F. Bancilhon, C. Delobel and P.Kanellakis (eds.) *Building an object-oriented database system. The story of O_2.* Morgan Kaufman Publishers, San Mateo, CA, 1992.

[17] Karlapalem, K. and Li, O. Partitioning Schemes for Object Oriented Databases. *Technical report.* University of Science and Technology, Department of Computer Science Clear Water Bay, Kowloon, Hong Kong. August, 1994a.

[18] Karlapalem, K., Navathe, A.B., and Morsi, M. Issues in Distribution Design in M.T.Ozsu, U.Dayal, and P.Valduriez (eds.) *Distributed Object Management.* Morgan Kaufman Publishers, San Mateo, 1994b.

[19] Khoshafian, S. *Object-Oriented Databases.* John Wiley & Sons, Inc. New York, 1993.

[20] Kim, H. Algorithmic and Computational Aspects of OODB Schema Design in R. Gupta, and E. Horowitz (eds.) *Object-Oriented Database with Applications to Case, Networks and VLSI CAD.* Prentice Hall Series in Data and Knowledge Base Systems, Englewood, NJ, 1991.

[21] Kim, W. *Introduction to Object-Oriented Databases.* MIT Press, Cambridge, MA, 1990.

[22] Kim, W., Garza, J.F., Ballou, N., and Woelk, D. Architecture of the ORION Next-Gereration Database Management System in M.Stonebraker (ed.) *Readings in Databse Systems.* Morgan Kaufman Publishers, San Francisco, CA, 1994.

[23] Lamg, C., Landis, G., Orenstein, J., and Weireb, D. The ObjectStore Database System in in M.Stonebraker (ed.) *Readings in Databse Systems.* Morgan Kaufman Publishers, San Francisco, CA, 1994.

[24] Lecuse, C., Richard, P., and Velez, M. An Object-Oriented Data Model in F. Bancilhon, C. Delobel, and P.Kanellakis (eds.) *Building an Object-Oriented Database System. The story of O_2.* Morgan Kaufman Publishers, San Mateo, CA, 1992.

[25] Lin, X., Orlowska, M., and Zhang, Y. A Graph Based Cluster Approach for Vertical Partitioning in Database System. *Data & Knowledge Engineering*, Vol. 11, No.3, New York, October 1993.

[26] Masunaga, Y. Object Identity, Equality and Relational Concepts in W. Kim, J.M. Nicolas and S. Nishio (eds.) *Deductive and Object Oriented Databases.* North-Holland, Amsterdam, 1991.

[27] McCormick, W., Schweitzer, P., and White, T. Problem Decomposition and Data Organization by a Clustering Technique. *Operations Research*, Vol. 20, No.1, Whippany, NJ, September 1972.

[28] Navathe, S., Ceri, G., Wiederhold, G., and Dou, J. Vertical Partitioning Algorithm for Database Design. *ACM Transaction on Database System*, Vol.9, No.4, New York, December 1984.

[29] Navathe, S. and Ra, M. Vertical Partitioning for Database Design: A Graphical Algorithm. *ACM SIGMOD*, Portland, OR, June 1989.

[30] Özsu, M.T., and Valduriez, P. *Principles of Distributed Database System.* Prentice Hall, Englewood Cliffs, NJ, 1991.

[31] Pernul, G., Karlapalem, K., and Navathe, S.B. Relational Database Organization Based on Views and Fragments. *Proceedings of the Second International Conference on Data and Expert Systems Applications*, Los Angeles, CA, 1991.

[32] Stonebraker, M. Inclusion of New Types in Relational Data Base Systems in M.Stonebraker (ed.) *Readings in Databse Systems.* Morgan Kaufman Publishers, San Francisco, CA, 1994.

BIOGRAPHICAL SKETCH

Elzbieta Malinowski was born in Glogow, Poland. She received her Master of Science in Engineering degree, with distinction, in 1982 from the Leningrad V.I.Ulyanov (Lenin) Electrical Engineering Institute in the former Soviet Union.

Since 1985 she has been employed as a professor of computer science in the Department of Computer and Information Science at the University of Costa Rica.

The fall of 1994 she joined the Department of Computer and Information Sciences at the University of Florida in pursuit of her graduate studies. She will receive her Master of Science degree in computer and information sciences from the University of Florida, Gainesville, in August 1996. Her post-graduate plans are to continue working at the University of Costa Rica.

Her research interests include distributed relational and object-oriented databases.