**CRAIOVA UNIVERSITY**
**FACULTY OF AUTOMATICS, COMPUTERS AND ELECTRONICS**

# PHD THESIS

## ALGORITHMS FOR INCREASING THE SPEED OF INFORMATION TRANSFER IN DISTRIBUTED DATABASES

- ABSTRACT -

**PhD**
**ADRIAN RUNCEANU**

**Coordinator:**
**Professor Ph.D. Eng. MIRCEA PETRESCU**

**2009**

This thesis presents generally several implementation problems about data fragmentation and in particular about the design of vertical partitioning in distributed relational databases and for distributed object-oriented databases and also horizontal fragmentation with the data allocation in the distributed object-oriented databases.

1. Thus it was studied the applying of several clustering algorithms for the distributed databases design. Some algorithms, such as error-squares algorithm, were adapted and implemented for data vertical fragmentation, too.

2. It was presented an objective function for n partitions, composed of two terms which provides a good process for decreasing the cost of transactions.

3. The function can be modified by adding new information about the types of queries (for data update or data retrieval), information relating to the allocation of data in the network and the cost of data access located at a distance.

4. The fragments allocation presented through an heuristical approach propose - in the case of distributed object-oriented databases - a variant that combines the classes instances fragmentation at the same time as the data allocation on different sites.

The PhD thesis is structured on 7 chapters, followed by the annexes and bibliography.

The **Chapter 1 "Introduction"** presents the main aspects of the distributed databases design.

The distributed databases design is an optimization process that requires to obtain solutions to several interpenetrating problems, namely data fragmentation, data allocation and local optimization. Each problem can be solved through different approaches, and therefore the design of distributed databases is becoming a very difficult task. Usually the design process is by definition heuristical.

The distributed databases design derived from the non-distributed databases design only in terms of distribution. The design involves data acquisition, database partitioning, the allocation and replication of partitions and local optimization. The database partitioning can be performed in several different ways: vertical, horizontal and hybrid (also called mixed) partitioning. This aspect of a database design will be highlighted in this thesis and namely the developing of an algorithm for checking some various algorithms proposed in literature for distributed databases partitioning (fragmentation).

Basically, it will be considered the problem of vertical data partitioning (fragmentation), also known as the attributes partitioning. This technique is used for the databases design to improve the performance of transactions. In vertical partitioning, the attributes of a relation R are together in groups that do not overlap and the relationship R is designed on relations fragments in accordance with these attributes groups.

In the distributed databases, these fragments are allocated on different sites. Here comes the aim of vertical partitioning to create vertical fragments of a relationship to minimize the cost of the data accessing during the transaction process. If the fragments are closer as possible to the needs of the transactions set, then the transactions processing cost can be reduced.

For the distributed databases design, the transaction cost is reduced by increasing the local transactions (on a site) and at the same time by reducing the amount of data views which are not local. The aim of vertical partitioning technique (and in general, partitioning techniques) is to find a partitioning scheme to meet the objective outlined above.

Note that the partitioning problem can be tackled on different levels of detail considering some additional information.

**The data fragmentation**. In this thesis, I have to consider only the information about transactions as input data to manage the partitioning problem effectively. In fact, the global optimization problem (which includes a large number of parameters and a metric complex) is divided into several smaller optimization problems in order to reduce the search space and to reduce the complexity of each problem separately.

In specialty literature are proposed several vertical partitioning algorithms, so it can be measured the affinity between pairs of attributes and try to group attributes together under the affinity between, using the algorithm BEA (bond energy algorithm). In an article we use an heuristical cost estimator for design a file to obtain a partitioning scheme "bottom-to-top". In another article extends the approach proposed by BEA and algorithm has two phases for vertical partitioning. The partitioning algorithms mentioned above use several heuristical methods for a relationship fragmentation.

As input date for these algorithms it is used the attributes matrix (AUM). This matrix has the attributes as columns, the transactions as lines and the transactions frequency access are values in the matrix. The most previous algorithms used for the data fragmentation use of affinity matrix of attributes (AAM) which is derived from AUM. AAM is a matrix in which for each pair of attributes it is stored the total frequency access of the transactions that access the pair of attributes. The results of different algorithms are often different, even if it is the same affinity matrix of the attributes, so it indicates that the objective functions used are different. Most vertical partitioning algorithms do not have an objective basis for assessing the correctness of partitions that are obtained through the application of those algorithms. Also, there is a common criterion or objective function to evaluate the results of these algorithms for vertical partitioning.

**Chapter 2 "A few basic concepts"** presents some basic concepts relating to distributed databases. Such are reviewed the models of representation for relational databases, distributed databases and object-oriented databases.

So, are described generally the databases systems, focusing on the components of their software. Then it is presented the relational data model and some operators of the structured query language (SQL).

The most popular database models so far are the relational model and the model that has managed to impose in recent years as a very good model, namely object-oriented model. All of them are presented in detail.

In recent years, the distributed databases (BDD) have become an important sector of processing information and it can be anticipated that their importance will grow rapidly. This trend is motivated both organizational and technological because BDD removes many of the shortcomings of centralized databases and are well received for the decentralizing organizational structures.

*A BDD can be defined logically like a collection of integrated data but physically distributed on the stations of computers network.*

This definition highlights two important aspects of BDD:

1. **The logical integrating of data collections**. For the user it is a single database with which the user interacts like with centralized databases (BDC).

2. **Physical distribution** refers to the database physical partitioning on the spaces from a computers network.

Both issues are vague enough to make differences between BDD and a set of local databases. During normal operations, the requests for applications from terminals of each station needs only the access to local database. These applications are executed only by the computer station and they are called **local applications**.

What distinguishes a set of local databases of a BDD is the existence of applications that access data from more stations. These applications are called global applications or distributed applications. It may comprise considerations that we have done in a working definition.

A BDD is a collection of logical integrated data that are physically distributed on stations of a computers network. Each station from the network has processing autonomy that enables it to achieve local applications. Also, each station participate on the implementation of global applications that require accessing data from more stations.

**Centralized control**. The ability to achieve a centralized control over the information resources of an enterprise or organizational structures has been considered one of the strongest motivations for entering the databases. The fundamental function of the database administrator is to ensure the protection and security for data, the data themself are recognized as important investment of enterprises that requires centralized control.

For the BDD the notion of centralized control is a little stressed. This depends on the architecture, too. Generally, we can identify a hierarchical control structure based on a **global manager** who has overall responsibility of the entire BDD and on the local administrators on their respective responsibilities linked to the local databases. Local administrators may have a high degree of autonomy which can go up to the coordination between the stations. BDD differ greatly in terms of autonomy from the complet autonomy of stations, without the global administrator, to the most complete centralized control.

**Data independence**. On the BDD the insurance of data independence from applications programs has the same significance as for BDC, but there is a new aspect of **distribution transparency**. By the distribution transparency the programs can be written notwithstanding the physical distribution of data. Moving data from one station to another only affects the speed of execution, not the correctness of the program.

As the data independence, the distribution transparency is obtained through a multi-architecture with different descriptions of the data.

**The insurance of a minimal and controlled redundancy**. In the BDD, there are several reasons to consider the data redundancy a desirable feature:

- the localization is faster when the data are replicated on all the stations where they are required by the applications;
- the system availability and security increases when the data are replicated. If a station fall the applications may be directed to the stations where the data are replicated.

The data redundancy reduces the effort for the data retrieval but increase the upgrade effort.

The evaluation of an optimal degree of redundancy must consider the report between data retrieval accesses and data update accesses.

**The integrity and restoring of data and the competition control**. In the BDD case, the integrity, restoring of data and the competition control achievements, although it relates to different issues, they are strongly interconnected. To a large extent, the solutions to these problems are related on the way of transactions achieving. A **transaction** is an atomic unit of execution, a sequence of operations that are either executed entirely or not performed. In the BDD, the problem of transactions atomicity becomes a particular issue related to the system behavior when one of the stations is not operational: to abort the whole transaction or to attempt a correct execution the transaction even if both stations are not simultaneously operational.

**The data safety and security**. In traditional databases, the database administrator who has the centralized control allows only authorized access to data. In BDD, the administrators face the same problems as traditional database administrators. They are mentioned two particular issues:
- In BDD, with a high degree of autonomy of the stations, the local databases are protected because the local administrators conducted their own protection not depend on a central administrator;
- Security issues are inherent to distributed systems because the network communication may be a weak point in achieving protection.

In **Chapter 3, "Architecture of distributed databases"** they are presented the main architectures of distributed databases, and **chapter 4" The database server"** offers for presentation the Oracle server for databases.

**Chapter 5, "The distributed databases fragmentation"** is the main chapter of this thesis because it presents more what does it means the data fragmentation if the databases are distributed, beginning with the fragmentation types and data fragmentation rules, then passing through the presentation of horizontal vertical and hybrid (mixed) fragmentations, and completing using clustering algorithms for data fragmentation, and particular case, both the vertical fragmentation in distributed and relational databases, as in the case of distributed object-oriented databases.

This chapter presents all the necessary concepts for understand and apply the data fragmentation in distributed databases, both in a relational and object-oriented ways.

Thus they are described the reasons for the using of fragmentation in the distributed databases design together with the known general types of fragmentation and exemplified by means of a database scheme. It is shown the fragmentation degree of a database and they are determined the fragmentation rules that must be followed in the distributed databases design.

About the distribution data, there is no need to fragment the data. After all, in the distributed files systems, the distribution is running on all the files. In fact, the data processing is done using files allocation in the network nodes.

In the fragmentation, the main feature is to close the units of distribution. A relationship is not a convenient unit for several reasons:

- First, the application views are usually included in the relations sets. Therefore, the localization of the applications access is not defined on all the relationships but on a part of them. So, it is natural to consider the subsets of relations as a ***distribution unit***.
- Secondly, if the applications that have views defined on a given relationship, are on different sites, then we can followe two alternative relationships with the entire unit of distribution:
  - First, the relationship is not replicated (copied) and stored on a single site;
  - On the other hand, it is replicated on all sites or on some sites where there are the applications.

The first variant requires a large and inefficient remotely access to the data. The second variant, on the other hand, provides a replication inefficient (unnecessary), causing problems in the updates implementation and might not be desirable if the storage capacity is limited.

Finally, the relationship decomposition into fragments, each treated as a unit, enables that some transaction to be running competitive.

In addition, the relations fragmentation results typically from the parallel execution in a single query by dividing into a multitude of subquerries that acts on fragments. Hence the fragmentation increases the competition.

We have to remember and the disadvantages of fragmentation. If applications have conflicting requirements that prevent the relationship decomposition into exclusive fragments, those applications whose views are defined on more than one fragment may suffer degradation in terms of performance. For example, it would be necessary to recover data from two fragments and it could get through union or through junction, which would be a pretty expensive operation. Avoiding this issue is an important feature of the fragmentation.

The second issue relates to the semantic data control, especially for the data integrity checking. As a result of the fragmentation, the participating attributes in the dependency can be broken down into different pieces which will be allocated on different sites. In this case, even the most simple act of dependencies checking will run by the data tracking on a number of sites.

**The horizontal fragmentation**. There are two types of horizontal fragmentation: primary fragmentation and derived fragmentation.

The factors that influence the fragmentation are: the structure of the database (global conceptual schema) and the application characteristics (the predicate used, the locations where they are stored, the frequency of access) - the "80/20" (20% of user querries use 80% of the data managed in the database).

The applications (which we can consider as querries), which access a database, select a data subset based on a predicate - that is a Boolean expression.

In the database design we should identify "groups" of tupls from the database that are accessed together by applications / queries. Each "group" is defined as a predicate "minterm" which is actually a combination of simple predicates. These "groups" then will become candidates for horizontal fragments

Quantitative information: ***minterm selectivity sel = (t)*** (the selectivity of a minterm mi) and ***frequency of access*** = ***acc (qi)*** (frequency of access query qi).

The horizontal partitioning algorithm is analyzed and outlined the steps to obtain necessary minterm predicates for horizontal fragments:

Step 1. Building a set of simple predicates to be complete and minimal.

Step 2. Generating set of minterm predicate.

Step 3. Elimination of some determined minterm predicates.

The derived horizontal fragmentation is defined on a member relationship of a link according to the selection operation specified on its own.

**The vertical fragmentation**. It is known that the vertical fragmentation of a relationship R produces the fragments R1, R2,. . ., Rn, each of them containing a subset of R's attributes as a primary key of R.

The objective of vertical fragmentation is to partition a relation into a subset of smaller relationships so that users applications to run only on a fragment.

There are two types of approaches in the vertical fragmentation on the global relations:

***Grouping***: *start by associating to each attribute a fragment, and at each step, the unification of fragments until the criterion is met.*

***The division (partitioning)*** *starts with a relationship and decides which partitions are useful, according to the applications for attributes accessing.*

**The information required for vertical fragmentation**. In design, the vertical fragmentation requires the following information: which applications / queries use attributes and the frequency with which these different applications / queries are running.

The first information will be stored into a matrix called the matrix created for the use of attributes.

The main information about the applications relates to the frequency of access.

Let Q = (Q1, Q2,. . ., Qq) be the set of user queries (applications) which runs on the relation R (A1, A2,..., An). Then, for every query qi and for each attribute Aj, we shall assign a value for the attribute using, denoted by use (qi, Aj), and defined as:

$$use(q_i, A_j) = \begin{cases} 1 & \text{if attribute } A_j \text{ is used by int query } q_i \\ 0 & \text{else} \end{cases}$$

The vectors use (qi, *) for each application are easy to define if the designer knows which applications will run on the database.

The values for the attributes use are not generally sufficient to form the basis for attributes sharing and fragmentation. This is because these values don't represent the difficult part of frequency applications. The frequency measurement can be included in the definition of measurement of attributes affinity, we can note it by aff (Ai, Aj), which measures the relationship between two attributes of a relationship in accordance when they are accessed by applications.

The measuring of affinity between two attributes Ai and Aj of a relation R (A1, A2,..., An) by the compliance of a set of applications Q = (Q1, Q2,. . ., Qq) is defined as

$$aff(A_i, A_j) = \sum_{k \mid use(q_k, A_i)=1 \ \wedge \ use(q_k, A_j)=1} \ \sum_{\forall S_l} ref_l(q_k)\, acc(q_k)$$

where:

- refl (qk) is the number of access on the attributes (Ai, Aj) for each running of the application qk on the site Sl
- and accl (qk) is the frequency of accessing the application defined as above but modified to include in the definition the frequency for different sites.

The clustering algorithm brings together the value attributes of greater affinities and separate those with lower affinity. The clustering algorithm receives as input the matrix AA, permutes lines or columns of the matrix (whatever it permutes because the matrix is symmetric) and generates the links affinity CA matrix (clustered affinity matrix). The permutation is made so as to maximize the global affinity measure.

The generating links affinity matrix (CA) is performed on three steps implemented by the algorithm GEN_CLUS_ATR:

***The initializing*** – it is placed and nominate an arbitrary column of the matrix AA into CA matrix (in the algorithm it was nominated column 1)

***The iteration*** - nominate a column between the rest of n-i columns (i is the number of columns already placed in CA) and it is placed on rest of i+1columns, it will be the place of the column that brings the greatest contribution to the global affinity described above. This step will continue until all the columns were placed.

**The lines ordering** – it is changed the positions of lines so as to be kept the symmetry. The partitioning algorithm aims the partitioning (division) that means to find a set of attributes that are free accessed (in most cases) by several distinct and various applications.

On the section 5.7 it is proposed an assessment of a data partitioning scheme in the distributed relational databases. It is proposed an algorithm for the assessment of the fragments results from the distributed database design stage and analize the algorithm on some proposed cases.

On every practical application for databases, a transaction does not require that all the usual attributes of a relationship tuples to be recovered during a transaction. When a relationship is vertically divided into data fragments the attributes stored in data fragments which are irrelevant (are not accessed by the transaction), they added a cost of recovery and processing, especially when the number of tuples involved in the relationship is very high.

In the distributed databases management systems, when a relevant attribute (an attribute that is accessed in a transaction) is in different data fragments that are allocated on different sites, then it is added an additional cost for access to these data. Therefore one of the desired characteristics for the distributed databases management systems that we want to reach through the partition, is the local accessible to any site. In other words, each site must be able to process local transactions with minimal access to the data located on other sites.

Ideally, we want for any transaction to access only those attributes from a single piece of data with minimal or no access to the relevant attributes of that segment. But it is impossible to reach this case since the transactions have different and overlapping access to subsets of attributes of a relationship. Moreover, the transactions are executed on different sites and some of the data fragments that contain the relevant attributes of a transaction can be found on sites located at a distance. The cost of transactions in a distributed environment consists of the cost of the local transaction and the cost of remote transactions. Even if it is possible to replicate data to avoid the cost of the process, for the first step we assume that we have no redundant data to avoid uperimposed and to ensure data integrity and consistency and also the cost of additional storage.

Suppose that during the databases design, the partitioning step is followed by the data fragments allocation when no superimposed data fragments obtained by partitioning are allocated on different sites, possibly with some replication. So the proposed evaluator will assess the the vertical partitioning schemes where the data fragments are no superimposed on the attributes ( it refers only to the primary key attributes). The primary key is reserved for each partition. It is necessary to obtain the original relationship without losing or adding tuples.

The aim of attributes partitioning and their allocation on different sites, has reached the minimal processing cost for any transaction initiated from any site. How the cost of transaction process has

two components, one relating to local processing and other remote processing, the partitioning assessor proposed that measure the "effectiveness" of the vertical partitioning scheme, also has two relevant terms: the term "the cost of local access to the irrelevant attributes", and the term "the cost of access to the relevant remote attributes".

For simplify, we assume that a single access to data fragments corresponds to a unit cost, this assumption can be easily relaxed if they are available more information about access methods, network parameters, etc.. The time-cost access to irrelevant attributes measure the cost of local processing transactions for the irrelevant attributes of the data fragments that are accessed remotly by the transactions; we can note that this contribution to the cost of remote access to irrelevant attributes is already included in the first term.

Since it is not known the data fragments allocation during the partitioning, it will be calculate the second term assuming that the data fragments necessary for a transaction are located on different sites. In the absence of any information relating to the strategies of transactions execution, we can calculate the second term or by determining the average of all costs of remote access obtained by executing the transaction on each site that contain a fragment necessary for the transaction or by the assumption that the transaction will run for the first time on the sites fragments. If it is available more information about the transactions strategies it can be introduced too in the second term.

The section 5.8 proposes, similar to the previous section, an evaluation of a partitiong scheme the data in the distributed object-oriented databases. The algorithm proposed in the previous section it is adapted to apply in the case of distributed object-oriented databases, by studying some cases according to the methods and specific attributes for the object-oriented approach.

On the relational approach the fragmentation works with algorithms that are appling to the attributes distribution. Depending on the type of fragmentation it is necessary information about the transaction frequency, the attributes use, or the predicate type. It is known that we need 4 information categories to reach an optimal design:

1. Information on the database that include global conceptual schema.
2. Information about the application: used predicate(for horizontal fragmentation), the matrix of used attributes and the transactions frequency (for vertical fragmentation).
3. Information about the network communication.
4. Information about computers system.

However, the object-oriented approach to conceptual schema may be more complex and different from features presented in relational approach, the additional aspects must are important, such are the methods, hierarchical structure and complex attributes.

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

- Finding an "affinity" between classes (all objects of a class represents an entity, we can not fragment the attributes and methods). "Affinity" between classes can be found in a similar manner as attributes of the relational approach.
- The classes and objects unification and finding the "affinity" between attributes.
- The methods distribution by using the access frequency on their specific transactions followed by the fragmentation of corresponding attributes accessed by those methods.
- Using an access frequency of the attributes for partitioning of a class objects followed by the methods distribution.
- Tracking a hierarchical structure of classes

In the developing process of a fragmentation model, we studied a single important situation, namely the use of simple attributes and simple methods.

In this category we have just a simple hierarchy, where the methods can use the attributes of their class and / or their superclass. We have nested methods or objects as a type attribute. I used other use matrices versus relational approache namely use transaction- method matrix - TMUM, use matrix method-attribute - MAUM. The use matrix transaction-method - TMUM is the

frequencies matrix that indicate if the transaction use certain methods. The values used are zero or one and they represent: ***use or not use*** the specified method. The use matrix method-attribute - MAUM is the matrix that represents the number of invoking of a specific attribute in a single execution of the method. There are three possible values:

- zero - indicate that a method does not access an attribute.
- one - indicate that a method read an attribute value (retrieve).
- two - indicate that a method read and write an attribute value (update).

In the proposed algorithm we must specify the following:

**Step 1.** To be specified into both matrices TMUM and MAUM all the methods used and the attributes from the root to the end of the hierarchical database structure where this algorithm was applied. If the classes from the higher levels exist, then for each of these classes we represent the access frequency of attributes and methods as a sum of corresponding values. If a class has subclasses, the amount will include all methods and attributes frequencies of that class and their subclasses.

**Step 2**. Multiply each TMUM line with the corresponding frequency value of transaction.

**Step 3.** Multiply TMUM with MAUM.

**Step 4**. Apply an exhaustive search and calculate the value of partitioning evaluator to select the best fragmentation scheme.

**Step 5**. Send the resulted fragmentation.

The aim attributes partitioning is to achieve a minimal cost process for a set of transactions and for using their attributes. It is unlikely to obtain an ideal fragmentation scheme, where each transaction accesses local only the necessary attributes and has no need to execute a remote access. The objective function proposed here attempts to balance the cost of local access and the remote access for a given transaction. According to the total cost of the transactions in a distributed environment consists of two elements:

1. The first component, named **the cost of access to local and irrelevant attributes** represents cost for accessing the irrelevant attributes when they access an object in a local site.

2. A second component, named **the cost of access to relevant to remote attributes** includes the cost for accessing relevant attributes by a transaction from sites located at a distance.

Chapter 6, "**The fragments assigning**" presents some ways for allocating data in distributed databases, and proposes an approach along the two-stage design of distributed databases, namely the fragmentation and allocation of data. The presented algorithm is analyzed together with a classical algorithm for horizontal fragmentation and it highlights based on some experimental evaluations, it is more advantageous in terms of designing a distributed database, to perform fragmentation data both horizontally and their allocation the network sites that are designed that distributed database.

Chapter 7 „Conclusions and further development " presents original contribution proposed in this phd thesis, and some ideas for further development and improvement of algorithms that are proposed in the thesis.

Personal contributions to this study in the approached field are:

The presentation of some classical algorithms for distributed databases fragmentation depending on different types of information fragmentation, namely, in the situation primary horizontal fragmentation it was presented the algorithm for horizontally partitioning, for the derived horizontal fragmentation I presented the algorithm for derived horizontal fragmentation.

On the other hand, we presented the algorithms BEA (Bond Energy Algorithm) and BVP (Vertical Partitioning Binary) for vertical fragmentation .

Proposing ways for assessing of the partitioning data scheme in a distributed relational database - EP algorithm.

Proposing ways of assessing the partitioning data scheme in a database distributed object-oriented - EPOO algorithm.

Proposing an heuristical approach for the horizontal fragmentation and allocation of fragments in the same phase of the design of databases distributed object-oriented. Comparative study between this approach and an algorithm proposed previously.

**The studying of the performance** of the algorithm proposed in a study on different data test and the analysis of the results with highlighting the fact that an algorithm is useful in optimizing the design of databases distributed relational fragmented vertically which should not be stated at the outset the total number of partitions. This study was presented at the International Joint Conferences on Computer, Information, Systems Sciences, and Engineering (CISSE 2007) Conference, Conference Proceedings book, 2007, University of Brigeport, USA, and published in Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering, Springer Science, ISBN 978-1-4020-8734-9 e-ISBN 978-1-4020-8735-6.

**The studying of the performance** of the algorithm proposed in other study on different test data and the analysis of the results with evidence that an algorithm is useful in optimizing the design of databases distributed object-oriented vertically fragmented which should not be stated at the outset the total number of partitions. In this paper we stressed that it is an adaptation of the algorithm in previous work, from databases to distributed relational databases distributed object-oriented. This study was presented and published at the 9th International Conference Carphatian Control Conference - ICCC 2008, Sinaia, ISBN 978-973-746-897-0.

The annexes include the distributed database schema used as the example throughout the thesis, followed by the the database structure used in experiments, and the component tables structure and the algorithms sources implemented in C + +.

In this thesis, I presented a general overview on the vertical data partitioning. This study highlights the possibility of using the affinity matrix of the attributes in partitioning algorithms. The use of an objective function from the grouping (clustering) methods to the partitioning algorithms it is presented by an effective implementation. Using this algorithm it can be evaluated and verified the other vertical partitioning algorithms that use as input a matrix for using attributes. This may develop heuristical algorithms that provide the possibility of using this objective function evaluation. This algorithm can be easily modified to include other information taken from the database designer, such as transactions types (data access or update data), information on data allocation and cost of data transmissions.

I proposed myself to develop further and others heuristical algorithms based on the approach proposed in this thesis and to integrate this algorithm into a more complex application that will help the distributed database designer to choose a proper way and the most effective implementation of information managed with a distributed database.