ГОДИШНИК НА МИННО-ГЕОЛОЖКИЯ УНИВЕРСИТЕТ "СВ. ИВАН РИЛСКИ", Том 50, Св. II, Добив и преработка на минерални суровини, 2007
ANNUAL OF THE UNIVERSITY OF MINING AND GEOLOGY "ST. IVAN RILSKI", Vol. 50, Part II, Mining and Mineral processing, 2007

# AN OBJECTIVE FUNCTION IMPLEMENTATION IN FRAGMENTATION DISTRIBUTED RELATIONAL DATABASES

***Adrian Runceanu***

*University Constantin Brâncuşi, Targu-Jiu, Romania; adrian_r@utgjiu.ro*

**ABSTRACT.** The design of distributed database design is an optimization problem and the resolution of several sub problems as data fragmentation (horizontal, vertical and hybrid), data allocation (with or without redundancy), optimization and allocation of operations (request transformation, selection of the best execution strategy, allocation of operations to sites). Each problem can be solved with some different approach this thing establishing that the project of the distributed databases to become hard enough. There are many researches connected by the dates fragmentation presented both in the case of relational database and in the case of orientated database. In this paper is presented the implementation of a heuristic algorithm conceived before that uses an objective function who takes over information about the administrated dates in a distributed database and it evaluates all the scheme of the database vertical fragmentation.
**Abbreviations**
Distributed databases, vertical fragmentation.

**ИМПЛЕМЕНТАЦИЯ НА КОНКРЕТНА ФУНКЦИЯ ПРИ ФРАГМЕНТИРАНЕТО НА СРОДНИ БАЗА ДАННИ**

***Адриан Рункеану***
*Университет „Константин Бранкуши", Търгу Жил, Румъния; adrian_r@utgjiu.ro*

**РЕЗЮМЕ.** Проектът за създаване на разпределена база от данни е оптимизационен проблем и решаване на някои подзадачи като фрагментиране на данните (хоризонтално, вертикално или хибридно), разпределение на данните (при наличие на повече или по-малко информация), оптимизиране и разпределяне на операциите (желана трансформация, избор на най-добре изпълнена стратегията, разпределението на дейностите по места). Всяка задача, която може да бъде решена по няколко различни начина е обстоятелство, което затруднява и осъществяването на замисъла за разпределение на базата данни става много труден. Научните изследвания, които съществуват, свързани с фрагментиране на информационни потоци, използват както случаите със сродни (релационни) база данни, така и с ориентирани база данни. Този доклад представя реализирането на евристичен алгоритъм, който е измислен преди използването на конкретна функция, която поема информация за управленски периоди в дадена база данни и отразява цялата схема на вертикална фрагментация на информационни данни.

## Previous work (introduction in vertical fragmentation of dates)

There exits three fragmentation types: vertical, horizontal and hybrid. Vertical fragmentation consists of subdividing a relation into sub relations which are projections of the original relation according to a subset of attributes. The horizontal fragmentation divides a relation into subsets of tuples based on selection operations. The hybrid fragmentation consists of dividing a relation horizontally, and then splitting vertically each of the obtained horizontal fragments or vice-versa.

Vertical fragmentation is used in order to increase transaction performance. The more obtained fragments are close to transaction requirements, the more the system is efficient. The ideal case occurs when each transaction matches exactly a fragment, i.e. it needs only this fragment. If some attributes are always used together, the fragmentation process is trivial. But in reality applications are rarely faced with such trivial cases. For relations having tens of attributes, it is necessary to develop systematic approaches for vertical partitioning. If a relation has m attributes, it can be partitioned following B(m) different ways, where B(m) is the $m^{th}$ Bell number which is almost $m^m$ [HN79].

Since the beginning of the 80's, many works have adressed the database vertical partitioning problem.

Navathe, Ceri and others [NCWD84] extend the work of Hoffer and Severance [HS75]. The authors use an attribute affinity matrix taht they order by using Bond Energy Algorithm as proposed in [HS75]. However, determining the vertical fragments is done automatically, whereas it was let the subjectif judgement of the designer in [HS75]. There are two steps in the partitioning algorithms. In the first step, the fragmentation is obtained by appying iteratively a binary partitioninf algorithm. At this step, no cost factor is considered. At second step, estimations of cost reflecting the physical environmemt, are included in order to optimise the initial fragments. The algorithm complexity is $O(n^2 logn)$, where n is number of attributes.

Ceri, Pernici and others [CPW89] propose two tools for vertical fragmentation: "DIVIDE" and "CONQUER". The tool "DIVIDE" performs only data fragmentation and allocation; it implements the partitioning algorithm proposed in [NCWD84]. The tool "CONQUER", in addition to data fragmentation and allocation, ensures the optimisation and allocation of operations.

Navathe and Ra proposed in 1989 a graphical tehnique of partitioning [NR89]. The attibute affinity matrix is considered as a complete graph where nodes represent attibutes and edges' weights represent the affinty values. The algorithm, by successively adding edges, generates all the fragments in one iteration by considering a cycle as a fragment. The algorithm has a complexity of $O(n^2)$, where n is number of attributes, and has the advantage of not using an objective function.

Lin, Orlowska and others [LOZ93] extend the work of [NR89] on graphical partitioning. The input to the algorithm is the affinity graph. They proposed searching a subgraph of at lest two nodes for which affinity values are greater than those of each incident edge.

Chakravarthy, Muthuraj and others [CMV94] have develop a partition evaluator which evaluates the partition quality by using two costs: the access cost to the irrelevant local attributes (present on the execution site of the transaction but not used by the transaction), and the access cost to the irrelevant remote attributes (not present on the execution site of the transaction but necesary for its execution).
This partition evaluator is implement in this paper.

## Input to the Vertical Partitioning Algorithm

The input to the Vertical Partitioning algorithm that we are going to explain is an Attribute Usage Matrix (AUM).

Algorithms such Bond Energy Algorithm, Binary Vertical Partitioning algorithm and Ra's Algorithms use the Attribute Affinity Matrix (AAM) formed from the Attribute Usage Matrix (AUM). Attribute affinity measures the bond between two attributes of a relation according to how they are accessed by applications. Attribute affinity between attribute i and j is given below:

$$Aff_{ij} = \sum_{i=1}^{T} q_{t,ij} \qquad (1)$$

where $q_{t,ij}$ is the number of accesses of transaction *t* referencing both attributes *i* and *j.*

The input assumed is a relation (consisting of a set of attributes) and an attribute usage matrix (AUM(t, j)) which consists of the attributes(j) in a relation as columns and the transactions(t) as rows with the frequency of access to the attributes for each transaction as the values in the matrix.

## Definitions and Notations

A *partition* (scheme) is a division of attributes of a relation into vertical fragments in which for any two fragments, the set of attributes of one is non-overlapping with the set of attributes of another. For example, the partition {(1, 3)(2, 4)(5)} defines a collection of fragments in which attributes 1 and 3 are in one fragment, 2 and 4 are in another and 5 is in a separate fragment. The following are used in the derivation of the Partition Evaluator.

*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 = I , 2 , . . . , 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 $f^i_{tj}$ is either 0 or $q_t$.
$A_{ij}$ : Attribute Vector for attribute *j* in fragment *i.. t-th* component of this vector is $f^i_{tj}$ .
$R_{itk}$ : 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*.

## Irrelevant local attribute access cost

For the first component we use square-error criterion as it was presented in [JD88]

The general objective is to obtain that partition which, for a fixed number of clusters, minimizes the square-error.
The square-error for the entire partition scheme containing *M* fragments is given by

$$E^2_M = \sum_{i=1}^{M} e^2_i \qquad (2)$$

## Relevant Remote Attribute Access Cost

Now we will include the second component which would compute a penalty factor that computes the function. Given a set of partitions, for each transaction running on a 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. This is summed over all the partitions and over all transactions giving the following equation. The second term is given by:

$$E^2_R = \sum_{t=1}^{T} \Delta_{i=1}^{M} \sum_{k \neq i} \left[ q^2_t * |R_{itk}| \frac{|R_{itk}|}{n^r_{itk}} \right] \qquad (3)$$

Here $\Delta^2$ is an operator that is either an average, minimum or maximum over all *i.* These different choices of the operator give rise to average, optimistic and pessimistic estimates of the remote access cost. If specific information is available regarding transaction execution strategies, then we can determine for each transaction *t,* the remote fragments accessed by the transaction and the remote access cost can be refined accordingly. In our experimental investigation, we use the optimistic estimate for illustration.

Partition Evaluator (PE) function is given by:

$$PE = E_M^2 + E_R^2 \quad (4)$$

*Analysis of the Partition Evaluator*

The final form of Partition Evaluator is given in equation 4. For analize and testing evaluator behavior, we implement an C++ program who produce all possible combinations of attribute with an number of fragments. We testing this program in several cases - an 5 attributes and 5 transactions, (1 to 10 fragments for case 1, 1 to 5 fragments for case 2), partition evaluator was computed, and for minimum values, partitions scheme was stored and write.

Program we used is composed from 2 algorithms, one (called PE algorithm) for computed value on a given partition scheme and an number of fragments, and the other algorithm (called GEN_EP algorithm) computed the minimal value of the PE from all partition schemes generated in a backtracking mode.

We present below values for each number of fragments and values for $E_M^2$ , $E_R^2$ and $EP$ .

For the test we used a matrice of attributes use with five attributes accesed by five transactions.

We present below values for each number of fragments together with the accordingly value opting for $E_M^2$ , $E_R^2$ and $EP$ .

We can notice that for a number of two fragments - the fragment I (1,4,5) and the fragment II (2,3) we obtain the lowest value for $EP$ .

We presented a general approach of the vertical fragmentation issue of the dates from a distributed database. Using an objective function used on the group models we obtained the implementation of an evaluator of partitions that can be use in the verification of some scheme of the dates fragmentation. We believe that now it's easier to project the euristic algorithm or other nature for the partition of databases. On the same principle it can be implementated a variant of this evaluator for the database oriented-object.

| *Transactions \ Attributes* | *A*1 | *A*2 | *A*3 | *A*4 | *A*5 |
|---|---|---|---|---|---|
| *T*1 | 0 | 30 | 0 | 30 | 30 |
| *T*2 | 15 | 15 | 15 | 0 | 15 |
| *T*3 | 40 | 0 | 0 | 40 | 40 |
| *T*4 | 0 | 10 | 10 | 0 | 0 |
| *T*5 | 15 | 15 | 15 | 0 | 0 |

| **Number of fragments** | **Partition scheme** | $E_M^2$ **values** | $E_R^2$ **values** | $EP$ **values** |
|---|---|---|---|---|
| 1 | (1,2,3,4,5) | 3477 | 0 | 3477 |
| 2 | (1,4,5) (2,3) | 1369 | 770 | 2139 |
| 3 | (1,4,5) (2) (3) | 791 | 1470 | 2261 |
| 4 | (1) (2) (3) (4,5) | 144 | 3192 | 3336 |
| 5 | (1) (2) (3) (4) (5) | 0 | 5836 | 5836 |

## References


[CPW89] S. Ceri, S. Pernici, and G. Weiderhold. Optimization Problems and Solution Methods in the Design of Data distribution. *Information Sciences Vol Uf, No. 3,* p 261-272, 1989.

[HS75] J. Hoffer, and D. Severance. The Uses of Cluster Analysis in Physical Database Design *In Proc. 1st International Conference on VLDB,* Framingham, MA, 1975, pp. 69 - 86.

[NCWD84] S. Navathe, S. Ceri, G. Wiederhold, and J. Dou. Vertical Partitioning Algo-for Database Design *ACM Transactions on Database Systems,* Vol. 9, Dec. 1984.

[NR89] S. Navathe, and M. Ra. Vertical Partitioning for Database Design: A Graphical Algorithm. *ACM SIGMOD*, Portland, June 1989.

[LOZ93] X. Lin, M. Orlowska, and Y. Zhang. A graph based cluster approach for vertical partitioning in database design. *Data an Knowlegde Engineering,* 11:151-169, 1993.

[CMVN94] S. Chakravarthy, R. Muthuraj, R. Varadarajan, and S. Navathe. An objective function for vertically partitioning relations in distributed databases and its analysis. In *Distributed and parallel databases*, pages 183-207. Kluwer Academic Publishers, 1994.

[JD88] A. Jain, and R. Dubes. Algorithms for clustering Data. Prentice Hall Advanced Reference Series, Englewood Cliffs, NJ, 1988.

[OV99] N. Tamer, P. Valduriez. Principles of Distributed Database Systems. Prentice Hall Englewood Cliffs, second edition, New Jersey 07362.


Recommended for publication by the Editorial board