TOWARDS VERTICAL FRAGMENTATION IN DISTRIBUTED DATABASES

Adrian Runceanu
University Constantin Brâncuşi Târgu-Jiu, adrian_r@utgjiu.ro

Abstract. The design of distributed database 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, and allocation of operations to sites). There are some different approaches to solve each problem, so this means that the design of the distributed databases is become hard enough. There are many researches connected to the dates fragmentation and they are presented both in the case of relational database and in the case of object-oriented 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.

Keywords: vertical fragmentation, distributed databases.

1. INTRODUCTION IN VERTICAL FRAGMENTATION OF DATABASES

There exist three fragmentation types: vertical, horizontal and hybrid. Vertical fragmentation consists of subdividing a relation into sub relations that 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. However, 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 (Hammer *et al.,* 1979).

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

(Hoffer and Severance*,* 1975) have developed the attribute affinity concept. This metric measures the frequency of accesing simultaneously a couple of attributes. The attributes having high affinity are grouped together by using the Bond Energy Algorithm developed by (Mc Cormick *et al.*. 1975).

(Hammer and Niamir, 1979) have proposed a heuristic where the input is a set of blocks corresponding each one to an attribute. This is the initial candidate partition. At each step of the search, several modifications of the partitions are generated and then submitted to a cost evaluator. If one of the modified partitions becomes the current candidate partition and the search continues until no modification is possible. Modifying a partition may be obtained in two different ways: by grouping two blocks or by regrouping an attribute i.e. by removing it from one bloc and inserting it into another one.

(Navathe, *et al.,* 1984) extend the work of (Hoffer and Severance*,* 1975). The authors use an attribute affinity matrix that they order by using Bond Energy Algorithm as proposed in (Hoffer and Severance*,* 1975). However, determining the vertical fragments is done automatically, whereas it was let the subjectif judgement of the designer in (Hoffer and Severance, 1975). 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.

(Cornell and Yu 1987) proposed a vertical partitioning algorithm which minimizes the number of disk accesses. The algorithm is based on integer programming methods. The partititoning of a relation requires the knowledge of several parameters concerning the relation (length, selectivity and number of attributes) and transaction types and behaviour (their frequency and the attributes they access).

(Ceri, *et al.* 1989) 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 (Navathe, *et al.,* 1984). 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. 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 *et al.* 1993) extend the work of (Navathe and Ra, 1989) 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, *et al.* 1994) 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).

Several authors have approached the generalization of fragmentation techniques to complex value and object oriented data models. For instance, horizontal fragmentation is discussed in (Bellatreche 2000, Karlapalem and Simonet 2000, Ezeife and Barker 1995, Ma 2003, Schewe 2002), and vertical fragmentation in (Chinchwadkar and Goh 1999, Ezeife and Barker 1998, Malinowski and Chakravarthy 1997, Schewe 2002).

2. STRUCTURE OF THE PAPER

In this paper we present the implementation of the partition evaluator which was describe in [CMV94].

We delimit our discussion to one of the data fragmentation problems, namely the vertical partitioning problem. Vertical Partitioning (also called attribute partitioning) is a technique that is used during the design of a database to improve the performance of transactions. In vertical partitioning, attributes of a relation R are clustered into non-overlapping groups and the relation R is projected into fragment relations according to these attribute groups. In distributed database systems, these fragments are allocated among the different sites. Thus the objective of vertical partitioning is to create vertical fragments of a relation so as to minimize the cost of accessing data items during transaction processing. If the fragments closely match the requirements of the set of transactions provided, then the transaction processing cost could be minimized. Vertical partitioning also has its use in partitioning individual files in centralized databases, and dividing data among different levels of memory hierarchies etc. In the case of distributed database design, transaction processing cost is minimized by increasing the local processing of transactions (at a site) as well as by reducing the amount of accesses to data items that are not local. The aim of vertical partitioning technique (and in general data partitioning techniques) is to find a partitioning scheme which would satisfy the above objective.

3. CONTRIBUTION

In this paper we are using the approach of formulating an objective function (named Partition Evaluator) before developing (heuristic) algorithms for the partitioning problem. This approach enables us to study the properties of algorithms with respect to an agreed upon objective function, and also to compare different algorithms for "goodness" using the same criteria. The objective function formulated in this paper is a step in this direction. Moreover, the objective function derived in this paper can be easily extended to include additional information (e. g., query types - retrieval/update, allocation information about the partitions, remote processing cost, and the transaction usage pattern at any particular site).

4. 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 = 1 , 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*.

Some algorithms such as Bond Energy (Hoffer and Severance, 1975), and (Navathe and Ra, 1989), use affinity matrix as the input. The attribute affinity is a measure of an imaginary bond between a pair of attributes. Because only a pair of attributes is involved, this measure does not reflect the closeness or affinity when more than two attributes are involved. Hence the algorithms which use attribute affinity matrix are using a measure (that is an ad hoc extrapolation of pair wise affinity to cluster affinity) that has no bearing on the affinity as measured with respect to the entire cluster. As a consequence, we believe, it was difficult to show or even characterize affinity values for the resulting clusters having more than two attributes.

As we wanted to obtain a general objective function and a criterion for describing affinity value for clusters of different sizes, our approach does not assume an attribute affinity matrix. The input model that we consider is a matrix which consists of attributes (columns) and the transactions (rows) with the frequency of access to the attributes for each transaction, as the values in the matrix. With this input model we overcome the limitations that are inherent to approaches based on attribute affinity matrix.

The objective function used by one algorithm is not suitable for evaluating the "goodness" of other algorithms. Thus we do not have a common objective function to compare and evaluate the results of these partitioning algorithms, or in general evaluate the "goodness" of a particular partitioning scheme. Hence we need a partition Evaluator to compare and evaluate different algorithms that use the same input in the database design process. Since attribute usage matrix is the most commonly used input available during the initial design stage, we first design an Evaluator which can be used to evaluate the "goodness" of partitions arrived at using this input. This Partition Evaluator can be used as a basis for developing algorithms to create fragments of a relation. With this approach, there is hope that admissibility aspects of algorithms can be shown. In addition, this Partition Evaluator has the flexibility to incorporate other information, such as type of queries (retrieval/updates), allocation information about the partitions, remote processing cost (transmission cost) and the transaction usage pattern at any particular site.

In any practical database application, a transaction does not usually require all the attributes of the tuples of a relation being retrieved during the processing of the transaction. When a relation is vertically divided into data fragments, the attributes stored in a data fragment that are irrelevant (i.e., not accessed by the transaction) with respect to a transaction, add to the retrieval and processing cost, especially when the number of tuples involved in the relation is very large. In a centralized database system with memory hierarchy, this will lead to too many accesses to the secondary storage. In a distributed database management system, when the relevant attributes (i.e., attributes accessed by a transaction) are in different data fragments and allocated to different sites, there is an additional cost due to remote access of data. Thus one of the desirable characteristics of a distributed database management systems that we wish to achieve through partitioning is the local accessibility at any site. In other words, each site must be able to process the transactions locally with minimal access to data located at remote sites.

4.1. Irrelevant local attribute access cost

For the first component we use square-error criterion as it was presented in Jain A. and Dubes R.. (1988).

The general objective is to obtain that partition which, for a fixed number of clusters, minimizes the square-error.

Let us assume that *n* attributes have been partitioned into *M* fragments (P_1, P_2, …, P_m) with n_i attributes in each fragment. Thus $\sum_{i=1}^{M} n_i = n.$ The mean vector V_i for fragment *i* is defined as follows.

This mean vector represents an average access pattern of the transactions over all attributes of fragment *i.* For an attribute vector A_{8J}, $(A_{ij} — V_i)$ is called the "difference vector" for attribute *j* in fragment *i.* The square-error for the fragment P_i - is the sum of the squares of the lengths of the difference vectors of all the attributes in fragment i. It is given by

$$e_i^2 = \sum_{j=1}^{n_i} (A_{ij} - V_i)^T (A_{ij} - V_i) \qquad 0 < i \le M \quad (3)$$

If $A_{ij} = V_i$ then e_i^2 will be zero. This will occur for the trivial case when there is a single attribute in each fragment or for the case when all the attribute in each fragment are relevant to all the transactions that access that fragment. It is the latter case that we are interested in and to avoid the former case, we will use the second component.

Transactions \ Attributes	$A1$	$A2$	$A3$	$A4$	$A5$
$T1$	0	30	0	30	30
$T2$	15	15	15	0	15
$T3$	40	0	0	40	40
$T4$	0	10	10	0	0
$T5$	15	15	15	0	0

The square-error for the entire partition scheme containing *M* fragments is given by

$$E_M^2 = \sum_{i=1}^{M} e_i^2 \quad (4)$$

4.2. 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:

$$E_R^2 = \sum_{t=1}^{T} \Delta_{i=1}^{M} \sum_{k \ne i} \left[q_t^2 * |R_{itk}| \frac{|R_{itk}|}{n_{itk}^r} \right] \quad (5)$$

Here Δ^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:

$$PE = E_M^2 + E_R^2 \quad (6)$$

5. ANALYSIS OF THE PARTITION EVALUATOR

The final form of Partition Evaluator is given in equation 6. For analize and testing evaluator behavior, we implement an C++ program who produce all possible combinations of attribute with an number of fragments. We test this program in three cases: case 1 - a 10 attributes and 8 transactions matrix; case 2 - a 5 attributes and 5 transactions, and case 3 – a 6 attributes and 4 transactions (1 to 10 fragments for case 1, 1 to 5 fragments for case 2, 1 to 4 fragments for case 3) 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_PE** algorithm) computed the minimal value of the PE from all partition schemes generated in a backtracking mode.

The algorithm on which base we implemented the evaluator of parts is presented below.

The outgoing data consists of the value for the local cost of access at irrelevant local attribute cost $E^2_{M,,}$ the access cost on the distance of the relevant attributes E^2_R, respective the value of the fragmentation evaluator - *EP*

First we implemented the algorithm EP based on the formula from equation 6 that calculates the value of EP, for a given fragmentation scheme so we used an entrance date: the matrix used for attributes - A; the lots of fragments on which it calculated the value of EP, the relation -R.

```
Algorithm PE
Input: A = attribute usage matrix;
       R = Relation ; F = fragments set
Output: $E^2_M$: irrelevant local attribute cost;
        $E^2_R$: relevant remote attribute cost;
        EP : partition evaluator value
Begin
 $E^2_M=0$
 for i from 1 to number_of_fragments do
  begin
   $e_i=0$
   for j from 1 to number of attributes from i fragment do
   { $X_{ij} – V_i$ – mean vector for j attribute form i fragment }
      $e_i = e_i + (X_{ij}-V_i)^T*(X_{ij}-V_i)$
   end_for
   $E^2_M = E^2_M + e_i$
```

```
end_for
$E^2_R=0$
for t from 1 to number of transactions do
begin
 minim =maxint
  for i from 1 to number of fragments do
    for k from 1 to number of fragments do
      begin
            if k $\neq$ i  then
                        begin
{ $R_{itk}$ – set of relevant attributes in fragment k accessed remotely with respect to fragment i by transaction t }
{ $n^{remote}_{itk}$ – number of relevant in fragment k accessed remotely with respect to fragment i by transaction t }
                        if exists attribute in matrix A who is from k fragment then
                  $E^2_R= E^2_R+ (f^t_k)^2 * | R_{it} | * | R_{it} | / n^{remote}_{itk})$
                  end_if
                  if $E^2_{R\ min}$ < minim then
                        minim = $E^2_{R\ min}$
                  end_if
          end_for
          $E^2_R = E^2_R + E^2_{R\ min}$
    end_for
end_for
EP = $E^2_M + E^2_R$
End.{Algorithm PE}
```

The second algoritm is presented below:

```
Algorithm GEN_PE
Input: A = attribute usage matrix;
Output :  lowest PE value
                  partition scheme coresponding to the lowest EP value
Begin
  minim=maxint
  for frag from 1 to number_of_fragments do
  { one partition scheme is generation for frag }
   pe = call PE( A, frag, F)
   if pe<minim then
        minim = pe
        number_fragment = frag
      G = F{set G is one copy of set F for corresponding value of minim}
   end_if
  end_for
  write number_fragment, set G and PE value
End. {Algorithm GEN_PE}
```

For the execution of one transaction, we know that if a transaction could be run at one fragment and that fragment haven't one single attribute accessed by that transaction, then transaction not be run on that fragment.

For the first test we used a matrice of attributes use with ten attributes accesed by eight transactions.

Tranzactions\Attributes	1	2	3	4	5	6	7	8	9	10
*T*1	25	0	0	0	25	0	25	0	0	0
*T*2	0	50	50	0	0	0	0	50	50	0
*T*3	0	0	0	25	0	25	0	0	0	25
*T*4	0	35	0	0	0	0	35	35	0	0
*T*5	25	25	25	0	25	0	25	25	25	0
*T*6	25	0	0	0	25	0	0	0	0	0
*T*7	0	0	25	0	0	0	0	0	25	0
*T*8	0	0	15	15	0	15	0	0	15	15

We present in Figure 1 the values for each number of fragments and the values for E^2_M , E^2_R and EP .

The total number of fragments evaluated was 115975. Optimal value (minimum) is obtained for 3 fragments – fragment I (1,5,7), fragment II (2,3,8,9) and fragment III (4,6,10).

The program used to generate all the combinations of ten attributes accessed by eight transactions offers three solutions (for five fragments) and two solutions (for eight fragments), having the same value for EP. However, the project of distributed database can choose which scheme of partition wishes to use it.

Number of fragments	**Partition scheme**	E^2_M **values**	E^2_R **values**	EP **values**
1	(1,2,3,4,5,6,7,8,9,10)	15085	0	15085
2	(1,4,5,6,7,10) (2,3,8,9)	7091	1366	8457
3	(1,5,7) (2,3,8,9) (4,6,10)	3312	2508	5820
4	(1,5) (2,3,8,9) (4,6,10) (7)	2078	3950	6028
5	(1,5) (2,3,8,9) (4,6) (7) (10) (1,5) (2,3,8,9) (4,10) (6) (7) (1,5) (2,3,8,9) (4) (6,10) (7)	2078	4800	6878
6	(1,5) (2,3,8,9) (4) (6) (7) (10)	2078	5650	7728
7	(1) (2,3,8,9) (4) (5) (6) (7) (10)	2078	6900	8978
8	(1) (2,8,9) (3) (4) (5) (6) (7) (10) (1) (2,3,8) (4) (5) (6) (7) (9) (10)	1386	10308	11694
9	(1) (2,8) (3) (4) (5) (6) (7) (9) (10)	0	14000	14000
10	(1) (2) (3) (4) (5) (6) (7) (8) (9) (10)	0	18350	18350

Fig. 1 Results of the first test

For the second 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^2_M , E^2_R 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 .

For the third test we used a matrice of attributes use with six attributes accesed by four transactions.

Number of fragments	**Partition scheme**	E^2_M **values**	E^2_R **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

Fig. 2. Results of the second test

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

No. of frag-ments	Partition scheme	E_M^2 values	E_R^2 values	EP values
1	(1,2,3,4,5,6)	24895	0	24895
2	(1,4) (2,3,5,6)	7565	55	7620
3	(1) (2,3,5,6) (4)	7565	276	7841
4	(1) (2) (3,5,6) (4)	5063	11336	16399
5	(1) (2) (3,6) (5) (4)	0	22492	22492
6	(1) (2) (3) (4) (5) (6)	0	40913	40913

Fig. 3. Results of the third test

We can notice that for a number of two fragments - the fragment I (1,4) and the fragment II (2,3,5,6) we obtain the lowest value for EP .

6. CONCLUSIONS

In this paper it is 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. Using this evaluator it's easier to project the heuristic algorithm or other nature for the partition of databases.

REFERENCES

Bellatreche, L., Karlapalem, K. and Simonet A., (2000), 'Algorithms and support for horizontal class partitioning in object-oriented databases', *Distributed and Parallel Databases* 8(2), 155–179.

Ceri S., Pernici S., and Weiderhold G. (1989) Optimization Problems and Solution Methods in the Design of Data distribution. *Information Sciences* ***Vol. no. 3,*** p 261-272.

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

Chinchwadkar, G. S. and Goh, A. (1999), 'An overview of vertical partitioning in object oriented databases', *The Computer Journal* 42(1).

Cornell D., and Yu P. (1987) A Vertical Partitioning Algorithm for Relational Databases. *Proc. Third International Conference on Data Engineering,* pp. 30-35.

Ezeife, C. I. and Barker, K. (1995), 'A comprehensive approach to horizontal class fragmentation in a distributed object based system', *Distributed and Parallel Databases* 3(3), 247–272.

Ezeife, C. I. and Barker, K. (1998), 'Distributed object based design: Vertical fragmentation of classes', *Distributed and Parallel Databases* 6(4), 317–350.

Hammer N. and Niamir B. (1979), A heuristic aproach to attribute partitioning. *In Proceedings ACM SIGMOD Int. Conf. on Management of Data,* (Boston, Mass.), ACM, New York.

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

Jain A. and Dubes R.. (1988) *Algorithms for clustering Data*. Prentice Hall Advanced Reference Series, Englewood Cliffs, NJ.

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

Ma, H. (2003), Distribution design in object orienteddatabases, Master's thesis, Massey University.

Malinowski, E. and Chakravarthy, S. (1997), Fragmentation techniques for distributing object-oriented databases, in *D. W. Embley & R. C. Goldstein, eds, 'Conceptual Modeling - ER '97'*, **Vol. 1331** of *Lecture Notes in Computer Science, Springer*, pp. 347–360.

Navathe S., Ceri S., Wiederhold G., and Dou J.(1984) Vertical Partitioning Algorithm for Database Design *ACM Transactions on Database Systems,* Vol. 9.

Navathe S. and Ra M. (1989) Vertical Partitioning for Database Design: A Graphical Algorithm. *ACM SIGMOD*, Portland.

Schewe, K.-D. (2002), Fragmentation of object oriented and semi-structured data, in H.-M. Haav & A. Kalja, eds, *'Databases and Information Systems II', Kluwer Academic Publishers*, pp. 1–14.

Tamer O. and Valduriez P.. (1999) *Principles of Distributed Database Systems*. Prentice Hall Englewood Cliffs, Second Edition, New Jersey 07362.