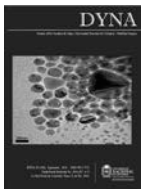
UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS



DYNA
http://dyna.medellin.unal.edu.co/

A hybrid partitioning method for multimedia databases

Lisbeth Rodríguez-Mazahua [a], Giner Alor-Hernández [a], Jair Cervantes [b], Asdrúbal López-Chau[c] & José Luis Sánchez-Cervantes [a]

[a] *Division of Research and Postgraduate Studies of the Instituto Tecnologico de Orizaba, México. lrodriguez@itorizaba.edu.mx, galor@itorizaba.edu.mx, isc.jolu@gmail.com*
[b] *Universidad Autónoma del Estado de México, Centro Universitario UAEM-Texcoco, México. chazarra17@gmail.com*
[c] *Universidad Autónoma del Estado de México, Centro Universitario UAEM-Zumpango, México. asdrubalchau@gmail.com*

Received: May 16th, 2015. Received in revised form: January 20th, 2016. Accepted: April 20th, 2016.

Abstract
Hybrid partitioning has been recognized as a technique to achieve query optimization in relational and object-oriented databases. Due to the increasing availability of multimedia applications, there is an interest in using partitioning techniques in multimedia databases in order to take advantage of the reduction in the number of pages required to answer a query and to minimize data exchange among sites. Nevertheless, until now only vertical and horizontal partitioning have been used in multimedia databases. This paper presents a hybrid partitioning method for multimedia databases. This method takes into account the size of the attributes and the selectivity of the predicates in order to generate hybrid partitioning schemes that reduce the execution cost of the queries. A cost model for evaluating hybrid partitioning schemes in distributed multimedia databases was developed. Experiments in a multimedia database benchmark were performed in order to demonstrate the efficiency of our approach.

Keywords: hybrid Partitioning; multimedia databases, query optimization.

Un método de fragmentación híbrida para bases de datos multimedia

Resumen
La fragmentación híbrida es una técnica reconocida para lograr la optimización de consultas tanto en bases de datos relacionales como en bases de datos orientadas a objetos. Debido a la creciente disponibilidad de aplicaciones multimedia, surgió el interés de utilizar técnicas de fragmentación en bases de datos multimedia para tomar ventaja de la reducción en el número de páginas requeridas para responder una consulta, así como de la minimización del intercambio de datos entre sitios. Sin embargo, hasta ahora sólo se ha utilizado fragmentación vertical y horizontal en estas bases de datos. Este artículo presenta un método de fragmentación híbrida para bases de datos multimedia. Este método toma en cuenta el tamaño de los atributos y la selectividad de los predicados para generar esquemas de fragmentación híbridos que reducen el costo de ejecución de las consultas. También, se desarrolla un modelo de costo para evaluar esquemas de fragmentación híbridos en bases de datos multimedia. Finalmente, se presentan algunos experimentos en una base de datos de prueba con el fin de demostrar la eficiencia del método de fragmentación propuesto.

Palabras clave: fragmentación híbrida; bases de datos multimedia, optimización de consultas.

1. Introduction

Query optimization to reduce response time or to avoid the excessive use of system resources has been an active research field over the past decades [1].

Hybrid partitioning is a database design technique to improve query performance. It divides a relation or table into subsets of attributes and tuples in order to minimize the irrelevant data accessed by the queries. Hybrid partitioning has been typically applied to traditional databases (relational or object-oriented databases) to achieve query optimization.

Vertical partitioning divides a table T into a set of fragments $fr_1, fr_2, ..., fr_n$, such that each fragment fr_i contains a subset of the attributes and the primary key of table T. In contrast, horizontal partitioning splits table T into a set of fragments $fr_1, fr_2, ..., fr_n$, where each fragment fr_i has a subset of tuples of T.

How to cite: Rodríguez-Mazahua, L., Alor-Hernández, G., Cervantes, J., López-Chau, A. & Sánchez-Cervantes, J.L., A hybrid partitioning method for multimedia databases DYNA 83 (198) pp. 59-67, 2016.

DYNA 83 (198), pp. pp. 59-67, Septiembre, 2016. Medellín. ISSN 0012-7353 Printed, ISSN 2346-2183 Online
DOI: http://dx.doi.org/10.15446/dyna.v83n198.50507

There are two versions of horizontal partitioning: *primary* and *derived*. *Primary horizontal partitioning* of a table is performed by using predicates that are defined on that table. On the other hand, *derived horizontal partitioning* divides a table according to the predicates that are defined on another table. In this work, only primary horizontal partitioning is considered.

Hybrid partitioning can be accomplished in one of three ways: first, by performing vertical partitioning and then horizontally partitioning the vertical partitions (called VH partitioning), or by first performing the horizontal partitioning and then vertically partitioning the horizontal partitions (called HV partitioning), or by directly taking into consideration the semantics of the transactions [2].

Currently, multimedia applications are highly available [3-5], such as audio/video on demand, digital libraries, electronic catalogues, among others. The rapid development of multimedia applications has created a huge volume of multimedia data, which has exponentially incremented from time to time [6]. A multimedia database is crucial in these applications in order to provide efficient data retrieval.

Distributed and parallel processing on database management systems (DBMS) may improve the performance of applications that manipulate large volumes of data. This may be accomplished by removing irrelevant data accessed during the execution of the queries and by reducing the data exchange among sites, which are the two main goals of the design of distributed databases [7]. Therefore, partitioning techniques have been used in multimedia databases to improve the performance of applications.

Nevertheless, only vertical or horizontal partitioning techniques have been considered by the literature until now. Vertical partitioning reduces the irrelevant attributes accessed by the queries, but all the multimedia objects are stored in a fragment. Many of the queries issued to the multimedia databases only require some objects from the database. In order to improve the performance of the queries in multimedia databases, it is necessary to reduce access to irrelevant attributes and irrelevant objects; this is achieved with hybrid partitioning. For this reason, in this paper we propose a method for hybrid partitioning in multimedia databases. First, our method develops horizontal partitioning and then vertical partitioning, so it is therefore an HV partitioning algorithm.

This paper is structured as follows: in Section 2, the state of the art of hybrid partitioning in traditional and multimedia databases is presented. In Section 3, the Multimedia Hybrid Partitioning (MHYP) algorithm is described. In Section 4, the proposed cost model for the evaluation of different hybrid partitioning schemes is explained. Section 5 shows the performance evaluation of the queries. Finally, Section 6 presents the conclusion and future lines of research.

2. State-of-the-art

In order to clarify the difference between the related work and our approach, we classify them into two classes that are described in the following subsections.

2.1. Hybrid partitioning methods for traditional databases

Most mixed or hybrid partitioning algorithms only consider traditional databases. In [2], algorithms to generate candidate vertical and horizontal fragmentation schemes and a methodology for distributed database design using these fragmentation schemes were proposed for relational databases. They applied vertical and horizontal fragmentation schemes together to form a grid. This grid that consisted of cells was then merged to form mixed fragments.

An analysis algorithm for assisting distribution designers in the fragmentation phase of object oriented databases was proposed in [8]. The analysis algorithm indicated the most adequate fragmentation technique (vertical, horizontal or mixed) for each class in the database schema. In [9] a strategy to carry out the fragmentation phase of the distribution design of object oriented databases was proposed. Their fragmentation strategy has three steps: 1) the analysis, 2) the vertical fragmentation phase, and 3) the horizontal fragmentation phase.

In [10] a UML-based model for mixed fragmentation was presented. They validated their model using a case study with the concepts of attribute usage matrix and predicate usage matrix. A genetic algorithm for mixed fragmentation in relational databases which provides an improvement over previous works which considered vertical and horizontal partitioning separately was discussed in [11,12]. Compared to attribute partitioning only method, the mixed fragmentation design method produced database cost savings up to 69%. A mixed partitioning approach for multi-tenant data schema was provided in [13]. Their approach made a good scalability in multi-tenant shared database, while it can meet the optimal partitioning and multi-division.

Problems of the aforementioned hybrid partitioning methods in applying them to multimedia databases are the following: 1) Some techniques [2,8-10,13] do not consider the size of the attributes in the vertical partitioning stage since multimedia databases tend to be highly varied sizes (e.g., it is not the same to access an id of 8 bytes as a video of 8 MB): it is necessary to take into account the size of the attributes; 2) Some methods [2, 10] are based on affinity, which is the sum of the frequency of the attributes or predicates that are accessed together by the queries. A cost-based method is better for multimedia databases since it can incorporate more information in the creation of a fragment, such as selectivity of the predicates and size of the attributes, as well as the frequency of the queries; 3) Some techniques [2,11,12] only consider the minimization of the number of disk accesses. It is important to also reduce the transportation cost (i.e., the data exchange among sites) in order to optimize the queries in multimedia databases.

The hybrid partitioning method for multimedia databases proposed in this paper solves these problems because it takes into account the size of the attributes, the selectivity of the predicates, and the frequency of the queries to get hybrid fragments, which reduce the number of disk accesses and the transportation cost of the queries.

2.2. Partitioning methods for multimedia databases

The partitioning algorithms that take into account multimedia data only perform vertical or horizontal partitioning. In [14], primary horizontal fragmentation in distributed multimedia databases is addressed. The authors´ partitioning strategy is based on low-level multimedia features.

Table 1.
Comparison between Some Partitioning Algorithms

Approach	Multimedia Data	Hybrid Partitioning
Navathe et al. [2]	No	Yes
Baião and Mattoso [8]	No	Yes
Jagannatha et al. [10]	No	Yes
Ng et al. [11]	No	Yes
Gorla et al. [12]	No	Yes
Li et al. [13]	No	Yes
Saad et al. [14]	Yes	No
Getahun et al. [15]	Yes	No
Chbeir and Laurent [16]	Yes	No
Fung et al. [18]	Yes	No
Rodríguez and Li [19]	Yes	No
Rodríguez et al. [20]	Yes	No
MHYP	Yes	Yes

Source: the authors

In [15], semantic-based predicates implication required in current fragmentation algorithms is addressed in order to partition multimedia data efficiently. In [16], a formal approach dedicated to multimedia query and predicate implication is discussed. In [17], a horizontal partitioning algorithm for multimedia databases, called MHPA, is presented. MHPA is based on hierarchical agglomerative clustering.

A vertical partitioning technique was applied in an e-Learning video database system in [18] to achieve efficient query execution. The disadvantage was that this vertical partitioning technique did not consider the transportation cost of multimedia objects over the nodes of the network or the size of the multimedia objects. A vertical partitioning algorithm for distributed multimedia databases, called MAVP (Multimedia Adaptable Vertical Partitioning), is provided in [19], which takes into account the size of the attributes in the partitioning process. In [20], a system for dynamic vertical partitioning of multimedia databases, called DYMOND (DYnamic Multimedia ON line Distribution), is presented. It uses active rules for the dynamic vertical partitioning process. In Table 1, we present a comparative analysis that summarizes the relevant contributions of all these related works.

As we can see in Table 1, the implementation of hybrid partitioning in multimedia databases has two problems: (a) current hybrid partitioning algorithms do not take into account multimedia data; (b) only vertical and horizontal partitioning algorithms for multimedia databases have been developed. These deficiencies can be improved by: (a) developing a hybrid partitioning algorithm for multimedia databases, and (b) proposing a cost model to evaluate hybrid multimedia databases´ partitioning schemes. This proposal tries to solve the aforementioned deficiencies.

3. Multimedia hybrid partitioning algorithm (MHYP)

In this section, the Multimedia Hybrid Partitioning Algorithm (MHYP) is described in detail. MHYP consists of two phases:

1. **Obtaining the horizontal fragments:** The predicates of the queries are analyzed in order to obtain the initial horizontal fragments. MHPA [17] is used to obtain the horizontal fragments.

2. **Generating the hybrid fragments:** MAVP [19] is used to vertically fragment the horizontal fragments obtained in the first phase. As a result this gives the hybrid partitioning scheme.

In order to clarify our approach, we present the following scenario of a simple multimedia database used to manage equipment in a machinery sell company. The database consists of a table named `EQUIPMENT` (*id*, *name*, *image*, *graphic*, *audio*, *video*) in which each tuple describes information about a specific piece of equipment, including its image, graphic, audio, and video objects. Information regarding 10,000 pieces of equipment (four different types) is stored: 2500 push mowers, 2500 string trimmers, 2500 chain saws, and 2500 water pumps. Let us also consider the following queries:

q_1:Find all chain saws images and graphics
q_2:Find name, audio and video with id "WP01"
q_3:Find all graphic, audio and video
q_4:Find all water pump images

Similarly to [16], we have considered that data that are stored in a table T can be defined by having two kinds of attributes: atomic and multimedia attributes. Also, we have assumed a fixed attribute set $U=A \cup M$, where:

- $A=\{A_1,A_2, \ldots, A_p\}$ and each A_i (i=1, 2, …, p) is an atomic attribute associate with a set of atomic values (such as strings and numbers, among others) called the domain of A_i and denoted by dom(A_i).
- $M=\{M_1,M_2, \ldots, M_q\}$ and each M_j (j=1, 2, …, q) is a multimedia attribute, associated with a set of complex values (represented as sets of values or vectors) called multimedia features (such as, color, texture, shape, to mention a few). The domain of M_j is denoted by $dom(M_j)$.

Thus, given a table T that is defined over U, tuples t in T are denoted as $\langle a_1,a_2, \ldots, a_p,m_1,m_2, \ldots, m_q\rangle$ where a_i is in $dom(A_i)$ $(1 \le i \le p)$ and m_j is in $dom(M_j)$ $(1 \le j \le q)$. Every a_i (respectively m_j) is denoted by $t.A_i$ (respectively $t.M_j$).

3.1. Horizontal partitioning process

In this section, we first explain the information requirements of the horizontal partitioning process and then we present the steps of the multimedia horizontal partitioning algorithm MHPA.

3.1.1. Information requirements of horizontal partitioning

Qualitative and quantitative information about queries is required in order to develop the horizontal partitioning process [7]. Fundamental qualitative information consists of predicates used in user queries. Similarly to [16], the multimedia queries used in our approach are conjunctive projection-selection queries over T of the form $\pi_X\sigma_C(T)$, where X is a non empty subset of U and C is a conjunction of atomic select predicates, i.e., C: $P_1 \wedge ... \wedge P_m$ is defined as follows:

DEFINITION 1: An atomic selection predicate P_j is an expression of the form $P_j=A_i\ \theta\ a$, where $A_i \in A$, $a \in dom(A)$ and $\theta = \{=, \le, \ge, <, >, \text{like}\}$.

Two sets are required in terms of quantitative information regarding user queries:

1. **Predicate selectivity**: number of tuples of the relation that would be accessed by a user query specified according to a given predicate. If $Pr=\{P_1, P_2, ..., P_m\}$ is a set of predicates, sel_i is the selectivity of the predicate P_i.

2. **Access frequency:** frequency with which user query access data. If $Q=\{q_1,q_2, ..., q_s\}$ is a set of user queries, f_k indicates the access frequency of query q_k in a given period.

3.1.2. The steps of the horizontal partitioning algorithm (MHPA)

Inputs: The table *T* is to be horizontally partitioned, and sest of queries with their frequencies are the input data of the MHPA.

Step 1: Determine the set of predicates *Pr* used by queries defined in the table *T*. These predicates are defined on a subset of attributes *A*'(*A*' $\subseteq$ *A*). As in [21], we call each element of *A*' a relevant predicate attribute.

The third query (q_3) does not have any predicate because the graphic, audio and video objects of all pieces of equipment are retrieved. Therefore, this query is not relevant for horizontal partitioning; this will be analyzed by the vertical partitioning algorithm. The predicates used by the queries q_1,q_2, q_4 in our running example are presented in Table 2.

Step 2: Build the predicate usage matrix (PUM) of table *T*. This matrix presents queries in rows and predicates in columns. In this matrix $PUM(q_k, P_i)$=1 if a query q_k uses a predicate P_i, otherwise it is 0. PUM also contains the fequency f_k of each query q_k and the selectivity sel_i of each predicate P_i. The PUM of our running example is shown in Table 3.

Step 3: Construct a partition tree. MHPA is based on a bottom-up approach. It first begins with single predicate fragments. It then, forms a new fragment by selecting and merging two of their fragments. This process is repeated until a fragment composed of all predicates is made. This kind of bottom-up approach generates a binary tree, which is called a partition tree (PT) [22]. Fig. 1 shows the PT of the table `EQUIPMENT` obtained by MHPA.

When two fragments are merged, the amount of remote tuples (i.e., tuples located in another fragment) accessed is decreased while the amount of irrelevant tuples accessed by the queries is increased. For example, in Fig. 1 we can observe that in the Step 0 each predicate is located in a different fragment. Therefore, in the first fragment there are 2500 tuples (the selectivity of P_1, i.e., sel_1), in the second one there are only 1 tuple (sel_2), and in the third fragment there are 2499 tuples (sel_3-sel_2). Query q_1 has to access the first fragment, which has the 2500 relevant tuples needed to answer the query. Therefore, it does not have to access any irrelevant tuple and any remote tuple. The same happens to query q_2. Nevertheless, query q_4 has to access the second and the third fragments in order to retrieve the tuples with the name "WATER PUMP". It has to access one remote tuple (assuming that the tuple with the id="WP01" is going to be transported from the second fragment to the third fragment).

If the predicates P_1 and P_2 are merged into a fragment (as in the Step 1 of Fig. 1), query q_1 now would have to access 1 irrelevant tuple and query q_2 would have to access 2500 irrelevant tuples.

Table 2.
Predicates used by queries

Q	*Pr*
q_1	P_1: *name="CHAIN SAW"*
q_2	P_2: *id="WP01"*
q_4	P_3: *name="WATER PUMP"*

Source: the authors

Table 3.
Predicate usage matrix.

Q/Pr	P_1	P_2	P_3	f_k
q_1	1	0	0	15
q_2	0	1	0	10
q_4	0	0	1	20
sel_i	2500	1	2500	

Source: the authors

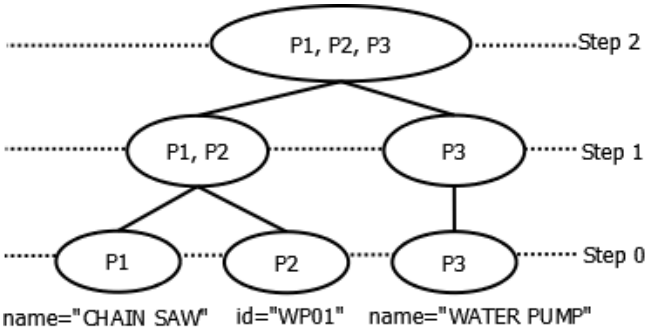


Figure 1. Partition tree of the `EQUIPMENT` table, obtained by MHPA.
Source: the authors

If P_1, P_2, and P_3 are merged (as in the Step 2) q_4 does not have to access 1 remote tuple (i.e., the tuple with the id "WP01" located in the second fragment). On the other hand, queries q_1 and q_4 now would have to access 2500 irrelevant tuples, and query q_2 now would access 4999 irrelevant tuples. Therefore, the merged fragment will increase the amount of accesses to irrelevant tuples and it will reduce the amount of access to remote tuples.

In MHPA, in each step during the construction of a PT, two nodes (fragments) are selected that maximize the merging profit that is defined below, when they are merged into a node (fragment).

$$Merging\_Profit(HPA) = DRT - \text{IIT} \qquad (1)$$

Where,

DRT: the decreased amount of remote tuples accessed.

IIT: the increased amount of irrelevant tuples accessed.

In each step during the construction of a PT, MHPA produces an horizontal partitioning scheme ps_i, which merges two fragments that maximize the merging profit function defined in equation 1. Therefore, when the PT is finished, we have a set of horizontal partitioning schemes $PS=\{ps_1, ps_2,..., ps_m\}$, and every ps_i has a set of fragments $ps_i=\{fr_1, fr_2,..., fr_i\}$.

To select two fragments of *i* fragments that can maximize the merging profit $\binom{i}{2} = \left(\frac{i(i-1)}{2}\right)$, pairs should be examined. For example in Step 0 ($ps_3 =\{fr_1, fr_2, fr_3\}$) of Fig. 1 i=m (where m is the number of predicates) because each predicate

Table 4.
Merging profits of the *EQUIPMENT* table in Step 0.

Pr	P_1	P_2	P_3
P_1		-25015	-87500
P_2			-25020
P_3			

Source: the authors

is located in a different fragment. Therefore, there are three fragments in Step 0 of Fig. 1 and it is necessary to examine the merging profits of $\left(\frac{3(3-1)}{2}\right)$ =3 pairs and merge one pair with the maximum merging profit among them. This generates the ps_2 of Step 1 in Fig. 1.

Table 4 shows MHPA Merging Profit Matrix (MPM) of the *EQUIPMENT* table in Step 0. In Algorithm 1, we show the process taken to get the MPM.

Algorithm 2 presents MHPA, it uses the PUM of the *T* and generates a set of initial horizontal fragments. Table 5 shows the horizontal partitioning schemes of the *EQUIPMENT* table that were obtained using MHPA.

Data: PUM of the table *T* (a set of predicates $Pr=\{P_1, P_2, ..., P_m\}$, the selectivity sel_i of each predicate P_i, a set of queries $Q=\{q_1, q_2, ..., q_s\}$, the frequency f_k of each query q_k}
Result: MPM: Merging Profit Matrix
for each $P_i \in Pr \mid 1 \leq i \leq m\text{-}1$ **do**
  for each $P_j \in Pr \mid i+1 \leq j \leq m$ **do**
    DRT=0;
    IIT=0;
    merging_profit=0;
    for each $q_k \in Q \mid 1 \leq k \leq s$ **do**
      if $PUM(q_k, P_i)=1$ & $PUM(q_k, P_j)=1$ **then**
        $DRT=DRT+f_k*(sel_i+sel_j)$;
      else
        if $PUM(q_k, P_i)=1$ **then**
          $IIT=IIT+f_k*sel_j$;
        else
          if $PUM(q_k, P_j)=1$ **then**
            $IIT=IIT+f_k*sel_i$;
          end
        end
      end
    end
    merging_profit=DRT-IIT;
    $MPM(P_i, P_j)$=*merging_profit*;
  end
end
Algorithm 1. getMPM

Data:PUM
Result: initial horizontal partitioning schemes $PS=\{ps_1, ps_2, ..., ps_m\}$
for each step ∈ PT **do**
  getMPM(*PUM*, *MPM*)
  select two nodes with maximum merging profit;
  merge the nodes;
end
Algorithm 2. MHPA

Table 5.
Resulting horizontal fragments of the table *EQUIPMENT*

PS	fr_1	fr_2	fr_3
ps_1	(P_1, P_2, P_3)		
ps_2	(P_1, P_2)	(P_3)	
ps_3	(P_1)	(P_2)	(P_3)

Source: the authors

Table 6.
Attribute usage matrix

Q/U	***id***	***name***	***image***	***graphic***	***audio***	***video***	f_k
q_1	0	1	1	1	0	0	15
q_2	1	1	0	0	1	1	10
q_3	0	0	0	1	1	1	25
q_4	0	1	1	0	0	0	20
s_i	8	20	900	500	4100	39518	

Source: the authors

3.2. Vertical Partitioning Process

MHYP uses the MAVP algorithm to achieve a vertical partitioning scheme (VPS). MAVP requires an Attribute Usage Matrix (AUM) as input, which has a set of atomic and multimedia attributes $U=A \cup M= \{ A_1, A_2, ..., A_p, M_1, M_2, ..., M_q \}$. The maximum size s_i of each attribute a_i is ∈ U, it has a set of queries $Q=\{q_1, q_2, ..., q_s\}$, the fequency f_k of each query is q_k, and it has a set of elements $AUM(q_k, a_i)$, where $AUM(q_k, a_i)$=1 if query q_k uses the attribute a_i,, or, if not, $AUM(q_k, a_i)$=0. The AUM of the *EQUIPMENT* table is presented in Table 6. MAVP takes into account the size of the attributes due to its importance in the vertical partitioning process because it is not the same to access a remote or irrelevant atomic attribute as it is to access a remote or irrelevant multimedia attribute. Multimedia attributes tend to be of a lot larger size. For further details, consider [19].

MAVP finds an optimal VPS when the number of fragments is equal to two $vps_2=\{fr_1=(id, audio, video), fr_2=(name, image, graphic)\}$.

3.3. Hybrid partitioning scheme generation

Algorithm 3 shows the MHYP algorithm. MHYP takes the PUM as an input as well as AUM of table *T* of the multimedia database and generates the optimal hybrid partitioning scheme (*optimal_hps*). Algorithm 3 presents the MHYP algorithm. MHYP obtains an optimial hybrid partitioning scheme (*optimal_hps*) based on the PUM and the AUM of the table *T*. MHPA uses the PUM to obtain a set of horizontal partitioning schemes $PS=\{ps_1, ps_2, ..., ps_m\}$. In contrast, MAVP only generates one optimal vertical partitioning scheme (VPS).

HPS_Generator combines the initial horizontal partitioning schemes generated by MHPA and the VPS obtained by MAVP. The number of hybrid partitioning schemes produced by the HPS_Generator is *m*, i.e., the number of horizontal partitioning schemes obtained by MHPA. Therefore, the HPS_Generator obtains a set of hybrid partitioning schemes $HPS=\{hps_1, hps_2, ..., hps_m\}$. Every hps_i has a set of fragments $hps_i=\{fr_1, fr_2, ..., fr_t\}$. Each fragment fr_k has n_k attributes. We suppose that the network has nodes $N_1, N_2, ..., N_t$, the allocation of the fragments to the nodes gives rise to a mapping $\lambda:\{1, ..., t\} \rightarrow \{1, ..., t\}$, which is called location assignment [23]. Table 7 depicts the definition of the hps_1 fragments.

HPS_Generator obtains two matrices: a Fragment-Attribute Usage Matrix (FAUM) and a Fragment-Predicate Usage Matrix (FPUM). FAUM contains a set of atomic and multimedia attributes $U=A \cup M= \{ A_1, A_2, ..., A_p, M_1, M_2,$

…., M_q }, the set of fragments of a hybrid partitioning scheme hps_i={fr_1, fr_2, ..., fr_t}, the sum of the size of the attributes sfr_k of each fragment $fr_{k,}$ and a set of elements FAUM(fr_k, a_i)=1 if fragment fragment fr_k has the attribute a_i, or, if not, FAUM(fr_k, a_i)=0. For instance, Table 8 shows the FAUM of the hps_1={fr_1, fr_2, fr_3, fr_4}, sfr_1=s_{id}+s_{audio}+s_{video}=8+4100+39518=43626. FPUM presents the fragments in rows and the predicates in columns. In this matrix, FAUM(fr_k, P_i)=1 if the fragment fr_k contains the tuples of the predicate P_i, or if not, it is 0. In addition, FPUM presents information about the cardinality cfr_k of a fragment fr_k. Table 9 presents the FPUM of the hps_1, every fragment stores 5000 tuples.

Data: *PUM, AUM*
Result: *optimal_hps*
best_cost=0;
optimal_hps=0;
MHPA(*PUM*, *PS*);
MAVP(*AUM*, *VPS*);
HPS_Generator(*PS*, *VPS*, *HPS*, *FAUM*, *FPUM*);
for each $hps_i \in$ HPS **do**
  cost(hps_i)=*IDAC*(hps_i)+*TC*(hps_i);
  if cost(hps_i)<*best_cost* **then**
    best_cost=cost(hps_i);
    optimal_hps=i;
  end
end
Algorithm 3. MHYP

4. Cost model

The cost of a hps_i is composed of two parts: irrelevant data access cost and transportation cost.

Table 7.
Fragments of the first hybrid partitioning scheme

hps_1
$fr_1=\pi_{a1, a5, a6}(\sigma_{P1\vee P2\vee P3}(T))$
$fr_2=\pi_{a1, a5, a6}(\sigma_{\neg P1\wedge\neg P2\wedge\neg P3}(T))$
$fr_3=\pi_{a1, a2, a3, a4}(\sigma_{P1\vee P2\vee P3}(T))$
$fr_4=\pi_{a1, a2, a3, a4}(\sigma_{\neg P1\wedge\neg P2\wedge\neg P3}(T))$
a_1=*id*, a_2=*name*, a_3=*image*, a_4=*graphic*, a_5=*audio*, a_6=*video*, T=*EQUIPMENT*

Source: the authors

Table 8.
FAUM

hps_1	*id*	*name*	*image*	*graphic*	*audio*	*video*	sfr_k
fr_1	1	0	0	0	1	1	43626
fr_2	1	0	0	0	1	1	43626
fr_3	1	1	1	1	0	0	1428
fr_4	1	1	1	1	0	0	1428

Source: the authors

Table 9.
FPUM

hps_1	P_1	P_2	P_3	cfr_k
fr_1	1	1	1	5000
fr_2	0	0	0	5000
fr_3	1	1	1	5000
fr_4	0	0	0	5000

Source: the authors

$$\boldsymbol{cost(hps_i) = IDAC(hps_i) + TC(hps_i)} \quad (2)$$

IDAC measures the amount of data from both irrelevant attributes and irrelevant tuples accessed during the queries. The transportation cost provides a measure for transporting between the nodes of the network.

The irrelevant data access cost is given by:

$$\boldsymbol{IDAC(hps_i) = \sum_{k=1}^{t} IDAC(fr_k).} \quad (3)$$

In order to obtain the cost of an hps_i, it is necessary to use the PUM and the AUM of a table *T*. The irrelevant data access cost of each hybrid fragment fr_k is given by:

$$\boldsymbol{IDAC(fr_k) = IAAC(fr_k) + ITAC(fr_k).} \quad (4)$$

IAAC is the irrelevant attribute access cost. ITAC is the irrelevant tuple access cost. IAAC is defined as:

$$\boldsymbol{IAAC(fr_k) = \sum_{q_j \in IQ_k} IAAC(q_j);} \quad (5)$$

$$\boldsymbol{IAAC(q_j) = \begin{cases} f_j \sum_{a_i \in IA_j} s_i \sum_{P_l \in PQ_j} sel_l & \text{if } q_j \in PUM; \\ f_j \sum_{a_i \in IA_j} s_i\, cfr_k & \text{otherwise} \end{cases}} \quad (6)$$

where IQ_k is a set of queries that uses at least one attribute and accesses at least one irrelevant attribute of the fragment fr_k. This is:

$$\boldsymbol{IQ_k = \{q_j | AUM(q_j, a_i) = 0 \wedge AUM(q_j, a_l) = 1 \wedge \{a_i, a_l\} \in fr_k\}} \quad (7)$$

$$\boldsymbol{\{a_i, a_l\} \in fr_k \Rightarrow FAUM(fr_k, a_i) = 1 \wedge FAUM(fr_k, a_l) = 1} \quad (8)$$

IA_j is the set of attributes that is not used by query q_j in IQ_k. This is defined as:

$$\boldsymbol{IA_j = \{a_i | AUM(q_j, a_i) = 0 \wedge FAUM(fr_k, a_i) = 1\}} \quad (9)$$

In the example IQ_1={q_3} because q_3 does not use the attribute *id* but it needs the attributes *audio* and *video* from the fragment fr_1 of the hps_1. Therefore, IA_3={*id*}.

PQ_j has the predicates used by a query q_j and is located in the fragment fr_k.

$$\boldsymbol{PQ_j = \{P_l | PUM(q_j, P_l) = 1 \wedge FPUM(fr_k, P_l) = 1\}} \quad (10)$$

For fr_1, PQ_1={P_1}, PQ_2={P_2}, PQ_3={∅}, PQ_4={P_3} because the predicates P_1, P_2, P_3 are accessed by the queries q_1, q_2, q_4 and the tuples required by the predicates are located in the fragment fr_1.

ITAC can be written as:

$$ITAC(fr_k) = \sum_{q_j \in AQ_k \wedge ITAC(q_j) \geq 0} ITAC(q_j); \quad (11)$$

$$ITAC(q_j) = \begin{cases} f_j\left(sfr_k\left(cfr_k - \sum_{P_l \in PQ_j} sel_l\right)\right) & if\ n_p \geq 1; \\ 0 & otherwise \end{cases} \quad (12)$$

where n_p is the number of predicates in PQj, and AQ_k contains the queries that access at leat one attribute of the fragment fr_k.

$$AQ_k = \{q_j | AUM(q_j, a_i) = 1 \wedge FAUM(fr_k, a_i) = 1\} \quad (13)$$

In the example for the fragment fr_1, $AQ_1=\{q_2, q_3\}$, since the query q_2 accesses all the attributes of the fragment fr_1 (*id*, *audio*, *video*) and q_3 accesses the attributes *audio* and *video* of fr_1. Due to the fact that $PQ_2=\{P_2\}$ and $PQ_3=\{\varnothing\}$, only the query q_2 contributes to the access to irrelevant tuples.

The transportation cost of an hps_i is computed according to a given location assignment. Since transportation costs dominate the execution cost of a query [7], the TC of hps_i is the sum of the costs of each query multiplied by its frequency squared, i.e.

$$TC(hps_i) = \sum_{j=1}^{t} TC(q_j)\ f_j^2 \quad (14)$$

The transportation cost of query q_j depends on the size of the relevant remote attributes and on the assigned locations, which decide the transportation cost factor between every pair of sites. It can be expressed by:

$$TC(q_j) = \sum_{h} \sum_{h'} c_{\lambda(h)\lambda(h')} s(h') sel(h') \quad (15)$$

where h ranges over the nodes of the network for q_j, $s(h')$, which are the sizes of the relevant remote attributes, $sel(h')$ is the number of relevant remote tuples accessed by the query, q_j, $\lambda(h)$ indicates the node in the network at which the query is stored, and c_{ij} is a transportation cost factor for data transportation from node Ni to node Nj {i,$j \in \{1, ..., t\}$}[23]. For instance, Table 10 presents the IAAC, ITAC, and IDAC of the hps_1 in megabytes. IAAC(fr_1)=IAAC(q_3), IAAC(q_3)=f_3*s_1*cfr_1=25*8*5000=1 million of bytes=1 MB. ITAC(fr_1)=f_2*sfr_1*(cfr_1-sel_2)=10*43626*(5000-1)=2180.86 MB.

The TC(hps_1) of MHYP is calculated as follows: there are four fragments, so we suppose that there are four nodes N_1, N_2, N_3, N_4, and each fragment fr_i is located in each node N_i. We also assume that each query is located in the node in which the larger attribute that it uses is located, and c_{ij}=1.

Table 10.
Costs of the first hybrid partitioning scheme

hps_1	IAAC	ITAC	IDAC
fr_1	1	2180.86	2181.86
fr_2	1	0	1
fr_3	141.71	196.34	338.05
fr_4	116.00	0.00	116.00

Source: the authors

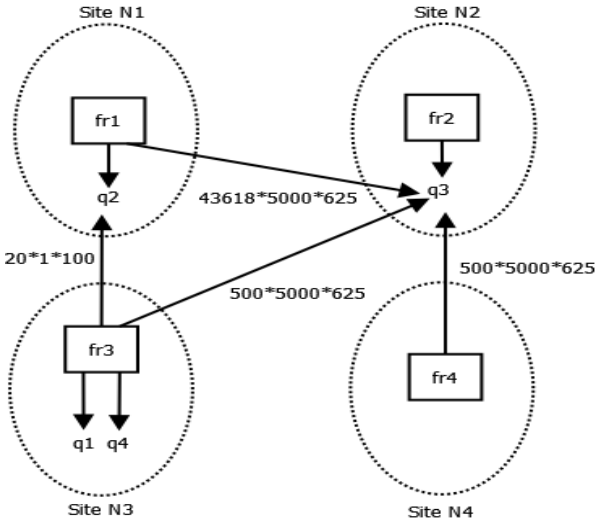


Figure 2. Location assignment of hps_1.
Source: the authors

Table 11.
Costs of the hybrid partitioning schemes

HPS	IDAC	TC	Cost
hps_1	2636.91	139431.25	142068.16
hps_2	1386.50	139431.62	140818.12
hps_3	259.70	139431.61	139691.31

Source: the authors

Fig. 2 illustrates the local assignment of hps_1. The query q_2 requires one tuple with the attribute *name*, which is located in the fragment fr_3. Therefore TC(q_2)=s_{name}* sel_2*f_2^2=20*1*10^2=2000 bytes.

Table 11 contains the costs of the hybrid partitioning schemes generated by MHYP. The optimal scheme is hps_3 and has a cost of 139691.31 MB.

5. Evaluation

This section presents and compares the hybrid partitioning schemes obtained using MHYP, the vertical partitioning scheme generated by MAVP, and the horizontal partitioning generated by MHPA. The benchmark used for the comparison was the database of a machinery sales company used in [19, 20] and described in Section 3. Some hybrid partitioning methods, such as [2, 11, 12] consider that the response time of a query is strongly affected by the amount of data accessed from secondary storage (disk). Hence, the objective functions of these methods are to minimize the number of disk accesses. The cost model proposed in this paper is used to compare the schemes obtained by MHPA, MAVP and MHYP since the cost to perform queries in distributed systems is dominated by the remote network communication as well as by local disk accesses.

Tables 12, 13 and 14 compare the costs of the queries of MHPA, MAVP and MHYP. As it can be observed, the scheme obtained with MHYP has a lower cost in most queries. This is because MHPA only takes into account

information about the irrelevant tuples accessed by queries, MAVP focuses on the reduction of irrelevant attributes, and MHYP considers the size of the irrelevant attributes and the selectivity of the predicates in order to reduce both irrelevant attributes and tuples accessed by the queries Using this information, MHYP considerably reduces the cost of the queries.

The cost of the query q_3 is increased in MHYP because it needs all graphic, audio and video of the multimedia database. In the scheme of MHPA, this query only accesses 5000 remote graphic, audio and video objects. The transportation cost of this query is considerably reduced in the MAVP scheme because it only has to access 10000 remote graphic objects. The scheme obtained by MHYP accesses 5000 remote audio and video objects and 10000 remote graphic objects, so its transportation cost is increased. Most of the queries executed in multimedia databases tend to access only a subset of attributes and tuples of the database; therefore, hybrid partitioning is suitable for these databases in order to reduce query execution cost.

6. Conclusion and future work

Hybrid partitioning optimizes query execution cost because it reduces the irrelevant data accessed by the queries. The novel aspects of our work include the following research contributions: first, a hybrid partitioning algorithm for distributed multimedia databases has been developed, which takes into account the size of the attributes and the selectivity of the predicates to generate an optimal hybrid partitioning scheme. Second, a cost model for distributed multimedia databases has been proposed. This cost model considers that the overall query processing cost in a distributed multimedia environment consists of irrelevant data access cost and transportation cost. An experimental evaluation shows that the algorithm proposed in this paper outperforms both a horizontal and a vertical partitioning only algorithm in most cases.

In this research we assumed that the queries that run against the multimedia database are static. Distributed multimedia databases are accessed by many users simultaneously, therefore queries tend to change over time and a good hybrid partitioning scheme can be degraded, resulting in very long query response time. Present research could be extended to derive the hybrid partitioning dynamically in multimedia databases (MMDBs) based on the changes in the queries. Thus, the hybrid partitioning scheme of the multimedia database can be adaptively modified to always achieve efficient retrieval of multimedia objects.

In the future, we also wish to consider low-level features of multimedia data and similarity-based (range and *k*-nearest neighbor) queries in the hybrid partitioning process. These kinds of queries are needed for content-based retrieval, which consists of obtaining information from the MMDB according to the characteristics of the multimedia objects, such as color, texture, and shape (in the case of images).

Table 12.
Comparison of the execution cost of the queries

MHPA			
Query	**IDAC**	**TC**	**Cost**
q_1	1635.97	0	1635.97
q_2	0.014	0	0.014
q_3	116	137868.75	137984.75
q_4	2205.42	0.368	2205.79

Source: the authors

Table 13.
Comparison of the execution cost of the queries

MAVP			
Query	**IDAC**	**TC**	**Cost**
q_1	0.31	0	0.31
q_2	4504.96	0	4504.96
q_3	234	3125	3359
q_4	239.6	0	239.6

Source: the authors

Table 14.
Comparison of the execution cost of the queries

MHYP			
Query	**IDAC**	**TC**	**Cost**
q_1	0.3	0	0.3
q_2	0.014	0	0.014
q_3	234	139431.24	139665.24
q_4	25.39	0.368	25.76

Source: the authors

Acknowledgments

The authors are very grateful to the Tecnológico Nacional de México for supporting this work. Also, this research paper was sponsored by the National Council of Science and Technology (CONACYT), as well as by the Public Education Secretary (SEP) through PRODEP.

References

[1] Moreno, F.J., Ospina-Romero G., yLarios-Restrepo R. Desempeño de consultas relacionales y objeto-relacionales en Oracle, Revista Ingeniería e Investigación, 25(3), pp. 4-12, 2005.

[2] Navathe, S., Karlapalem, K. and Ra, M., A mixed fragmentation methodology for initial distributed database design, Journal of Computer and Software Engineering, 3, pp. 1-34, 1995.

[3] Motato-Toro O.F. y Loaiza-Correa H., Identificación biométrica utilizando imágenes infrarrojas de la red vascular de la cara dorsal de la mano, Revista Ingenierìa e Investigación, 29(1), pp. 90-100, 2009.

[4] Atencio, P., Sánchez G.T. and Branch J.W., Automatic visual model for classification and measurement of quality of fruit: Case mangifera INDICA L, DYNA 76(160), pp. 317-326, 2009.

[5] Álvarez M.J, González E., Bianconi F., Armesto J. and Fernández A., Colour and texture features for image retrieval in granite industry, DYNA 77(161), pp. 121-130, 2010.

[6] Rahman, M.N.A., Lazim, Y.M., Mohamed, F., Saany, S.I.A. and Yusof M.K.M., Rules generation for multimedia data classifying usin rough sets theory, International Journal of Hybrid Information Technology, 6(5), pp. 209-218, 2013. DOI: 10.14257/ijhit.2013.6.5.19

[7] Özsu, M.T. and Valduriez, P., Principles of distributed database systems. New York: Springer, third edition, 2011.

[8] Baiäo, F. and Mattoso, M., Towards an inductive design of distributed object oriented databases, Proceedings of the Third IFCIS Conference of Cooperative Information Systems (CoopIS'98), New York, USA, IEEE CS Press, pp. 88-197, 1998.

[9] Baiäo, F., Mattoso, M. and Zaverucha, Z., A Distribution design methodology for object DBMS, Distributed and Parallel Databases, 16(1), pp. 45-90, 2004. DOI: 10.1023/B:DAPD.0000026268.04288.b9

[10] Jagannatha, S., Mrunalini, M., Kumar, T.V.S. and Kanth, K.R., Modeling of mixed fragmentation in distributed database using UML 2.0, Proceedings of the Int. Conf. on Computer Engineering and Applications, pp. 190-194, 2009.

[11] Ng, V., Gorla, N. and Law, D.M., Applying genetic algorithms in database partitioning, Proceedings of the 2003 ACM Symposium on Applied Computing (SAC), pp. 544-549, 2003. DOI: 10.1145/952532.952639

[12] Gorla, N., Ng, V. and Law, D.M., Improving database performance with a mixed fragmentation design, Journal of Intelligent Information Systems, 34, pp. 559-576, 2012. DOI: 10.1007/s10844-012-0203-x

[13] Li, H., Yang, D. and Zhang, X., A mixed partitioning approach for multi-tenant database approach, Journal of Information & Computational Science, 10(15), pp. 4869-4878, 2013. DOI: 10.12733/jics20102341

[14] Saad, S., Tekli J., Atnafu, S., Chbeir, R. and Yetongnon, K., Towards multimedia fragmentation, Advances in Databases and Information Systems, Lecture Notes in Computer Science, 4152, pp. 415-429, 2006. DOI: 10.1007/11827252_31

[15] Getahun, F., Tekli, J., Atnafu, S. and Chbeir, R., The use of semantic-based predicates implication to improve horizontal multimedia database, Proceedings of the MS'07 Workshop on Multimedia Information Retrieval on The Many Faces of Multimedia Semantics, New York, USA: ACM, pp. 29-39, 2007. DOI: 10.1145/1290067.1290073

[16] Chbeir, R. and Laurent, D., Towards a novel approach to multimedia data mixed fragmentation, Proceedings of the Int. Conf. on Management of Emergent Digital EcoSystems, New York, USA: ACM, pp. 200-204, 2009. DOI: 10.1145/1643823.1643860

[17] Rodríguez, M., Alor-Hernández, G., Abud-Figueroa M.A. and Peláez-Camarena S.G., Horizontal partitioning of multimedia databases using hierarchical agglomerative clustering, in: Gelbuk A. et al. (Eds.), MICAI 2014, Part II, LNAI 8857, Springer, pp. 296-309, 2014. DOI: 0.1007/978-3-319-13650-9_27

[18] Fung, C.-W., Leung, W.-C. and Li, Q., Efficient query execution techniques in a 4Dis video database system for eLearning. Multimedia Tools and Applications, 20(1), pp. 25-49, 2003. DOI: 10.1023/A:1023418316038

[19] Rodríguez, L. and Li, X., A vertical partitioning algorithm for distributed multimedia databases, in: Proceedings of DEXA 2011, 6861, Springer Verlag, pp. 544-558, 2011. DOI: 10.1007/978-3-642-23091-2_48

[20] Rodríguez, L., Li, X., Cervantes, J. and García-Lamont, F., DYMOND: An active system for dynamic vertical partitioning of multimedia databases, Proceedings of the 16th International Database Engineering & Applications Sysmposium, New York, USA: ACM, pp. 71-80, 2012. DOI: 10.1145/2351476.2351485

[21] Bellatreche, L., Karlapalem, K. and Simonet, A., Algorithms and support for horizontal class partitioning in object oriented databases, Distributed and Parallel Databases, 8, pp. 155-179, 2000. DOI: 10.1023/A:1008745624048

[22] Son, J.H. and Kim, M.H., An adaptable vertical partitioning method in distributed systems, Journal of Systems and Software, 73(3), pp. 551-561, 2004. DOI: 10.1016/j.jss.2003.04.002

[23] Ma, H., Distribution design for complex value databases. PhD Thesis, Massey University, Palmerston North, New Zeland, 2007.

L. Rodríguez-Mazahua, received her BSc in Information Technology and her MSc in Computer Science from the Instituto Tecnológico de Orizaba, Veracruz, Mexico, in 2004 and 2007, respectively. In 2012 she obtained a PhD in Computer Science from the Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV-IPN), Mexico. From 2012 to 2014, she was a professor of computer science at the Universidad Autónoma del Estado de México, Centro Universitario UAEM Texcoco, Mexico. Since February 2014 she has been undertaking postdoctoral research at the Instituto Tecnológico de Orizaba. Her current research interests include distribution design of databases, database theory, autonomic database systems, multimedia databases, and Big Data.
ORCID: 0000-0002-9861-3993

G. Alor-Hernández, is a full-time researcher at the Division of Research and Postgraduate Studies in Orizaba´s technological institute, the Tecnológico de Orizaba, Mexico. He received an MSc and a PhD in Computer Science from the Center for Research and Advanced Studies at the National Polytechnic Institute (CINVESTAV), Mexico. He has led 10 Mexican research projects granted by CONACYT, DGEST and PROMEP. He is author/coauthor of around 130 journal and conference papers on computer scienceHis research interests include Web services, e-commerce, Semantic Web, Web 2.0, service-oriented and event-driven architectures, and enterprise application integration. He is an IEEE and ACM Member. He is a National Researcher recognized by the National Council of Science & Technology of Mexico (CONACYT).
ORCID: 0000-0003-3296-0981, Scopus Author ID: 17433252100.

J. Cervantes, received his BSc. in Mechanical Engineering from Orizaba Technologic Institute, Veracruz, Mexico in 2001, his MSc. and PhD. from CINVESTAV-IPN, Mexico, in 2005 and 2009 respectively. His research interests include support vector machine, pattern classification, neural networks, fuzzy logic and clustering.
ORCID: 0000-0003-2012-8151, Scopus Author ID:23033927200

A. López-Chau, received his BSc. degree in Electronic Engineering from the Instituto Politécnico Nacional, México; his MSc. in computer science from the Centro de Investigación en Computación at the Instituto Politécnico Nacional, México, in 1997 and 2000, respectively. In 2013 he obtained a PhD. in Computer Science from CINVESTAV-IPN. Since 2011, he has been a professor of computer science at Universidad Autónoma del Estado de México, Centro Universitario UAEM Zumpango. His current research interests include data mining, machine learning and embedded systems.
ORCID: 0000-0001-5254-0939

J.L. Sánchez-Cervantes, obtained a PhD in Computer Science and Technology from the Universidad Carlos III de Madrid. He received an MSc. in Computer Systems and is an engineer in Computer Systems at the Instituto Tecnológico de Orizaba. His research interests include Semantic Web, Linked Data (Linked Open Data), Social Media, Big Data and Internet of things.
ORCID: 0000-0001-5194-1263