Towards Multimedia Fragmentation

Samir Saad[1], Joe Tekli[1], Richard Chbeir[1], Kokou Yetongnon[1]

[1] LE2I Laboratory UMR-CNRS, University of Bourgogne
21078 Dijon Cedex France
{samir.saad, joe.tekli}@khali. u-bourgogne.fr
{richard.chbeir, kokou.yetongnon}@ u-bourgogne.fr

Abstract. Database fragmentation is a process for reducing irrelevant data accesses by grouping data frequently accessed together in dedicated segments. In this paper, we address multimedia database fragmentation by extending existing fragmentation algorithms to take into account key characteristics of multimedia objects. We particularly discuss multimedia primary horizontal fragmentation and provide a partitioning strategy based on low-level multimedia features. Our approach particularly emphasizes the importance of multimedia predicates implications in optimizing multimedia fragments. To validate our approach, we have implemented a prototype computing multimedia predicates implications. Experimental results are satisfactory.

Keywords: Multimedia fragmentation, Range and KNN operators, predicates implication, objects classification

1 Introduction

Since the last two decades, multimedia data are of key importance in many application areas such as medicine, surveillance, cartography, meteorology, security, visual data communications, etc. Hence, the need for systems that can catalog, store, and efficiently retrieve relevant distributed multimedia data is becoming very high. Initially, research in multimedia management has been handled separately by database management and computer vision communities. As a result, different types of features have been used, in the literature, for multimedia data management. Low-level features such as color, texture, shape, layout, etc. are used by the computer vision research community, while meta-data and semantic based features are widely used by the database management community to describe data context and semantics. Emerging applications in distributed environments create an increasing demand on the performance of multimedia systems, requiring new data partitioning techniques to achieve high resource utilization and increased concurrency and parallelism. Several continuing studies are aimed at building distributed multimedia databases management systems MMDBMS [20]. Nevertheless, most existing systems lack a formal framework to adequately provide full-fledge multimedia operations.

Traditionally, partitioning techniques are used in distributed system design to reduce accesses to irrelevant data. Three main fragmentation techniques have been defined for relational databases: horizontal fragmentation HF, vertical fragmentation (VF), and hybrid or mixed fragmentation (MF). These techniques have been recently extended for object oriented databases. However, multimedia data fragmentation issues haven't been addressed in current systems.
Multimedia fragmentation is a relatively complicated issue owing to the complexity of the multimedia data itself; different multimedia data types (video, audio, image and/or text), frequently used with various formats, as well as the intricacy of the description of physical and/or semantic multimedia data. In this paper, we address primary horizontal fragmentation in distributed multimedia databases and analyze the impact of multimedia operators and predicates. We particularly address multimedia predicates implication required in current fragmentation algorithms such as Make_Partition and Com_Min [2, 11, 12]. We also present our prototype with corresponding experimental results conducted to validate our approach.
The remainder of this paper is organized as follows. Section 2 briefly reviews background in DB fragmentation. Section 3 presents a motivation example. Section 4 details our multimedia fragmentation process. Section 5 presents our prototype and experimental tests. Finally, section 6 concludes and draws future directions.

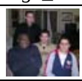
2 Background

Fragmentation techniques for distributed DB systems aim to achieve high resource utilization and performance [5]. This is addressed by removing irrelevant data accessed by applications and by reducing data exchange among sites [1]. In this section, we briefly present traditional database fragmentation approaches, depicting the evolution from relational to object oriented DBMS, and focus on horizontal fragmentation algorithms. In essence, there are three fundamental fragmentation strategies: Horizontal Fragmentation (HF), Vertical Fragmentation (VF) and Mixed Fragmentation (MF).
HF underlines the partitioning of an entity/class in segments of tuples/objects verifying certain criteria. The generated horizontal fragments have the same structure as the original entity/class. Horizontal fragmentation is generally categorized in two types: Primary HF and Derived HF. PHF is the partitioning of an entity based on its attributes' values [12]. DHF denotes the partitioning of an entity (called member) based on links with other entities (called owners) [12]. In other words, it is the partitioning of an entity/class in terms of the PHF of another entity/class [1] taking into consideration their inner-links.
VF breaks down the logical structure of an entity/class by distributing its attributes/methods over vertical fragments, which would contain the same tuples/objects with different attributes [1]. The unique tuple/object identifier (id) is kept in all vertical fragments [7] so that the DBMS can link related segments.
MF is a hybrid partitioning technique where horizontal and vertical fragmentations are simultaneously applied on an entity/class [11].
To the best of our knowledge, two main algorithms for the PHF of relational DBMS are provided in the literature: *Com_Min* developed by Oszu and Valduriez [12] and

Make_Partition Graphical Algorithm developed by Navathe *et al.* [10] (used essentially for vertical fragmentation). The *Com_Min* algorithm generates, from a set of simple predicates applied to a certain entity, a complete and minimal set of predicates used to determine the minterm fragments corresponding to that entity. A minterm is a conjunction of simple predicates [2] associated to a fragment. *Make_Partition* generates minterm fragments by grouping predicates having high affinity towards one another. The number of minterm fragments generated by *Make_Partition* is relatively smaller than the number of *Com_Min* minterm fragments [15] (the number of minterm fragments generated by *Com-Min* being exponential to the number of simple predicates considered).
Similarly, there are two main algorithms for the PHF of object oriented DBMS: one developed by Ezeife and Barker [6] using Com_Min [12], and the other developed by Bellatreche *et al.* [2] on the basis of Make_Partition [10]. The use of Com_Min or Make_Partition is the major difference between them.

3 Motivation

In order to use current partitioning approaches, widely employed in traditional databases, for fragmenting multimedia data, several issues should be studied and extended. On one hand, to achieve fragmentation, current algorithms require as an input parameter [6] the database conceptual schema (CS). This requirement is not always fulfilled in some multimedia databases due to the unstructured (or semi-structured) and complex nature of multimedia data. On the other hand, multimedia queries contain new operators handling low-level and semantic features. These new operators should be considered when studying predicates and particularly predicate implications. For example, let us consider the following predicates used to search for photos similar to given photos in an Employee multimedia database as shown below.

Predicate	**P1**	**P2**	**P3**	**P4**
Attribute	*Emp_photo*	*Emp_photo*	*Emp_photo*	*Emp_photo*
Operator[1]	*Range_Sim*$_{\varepsilon 1}$	*Range_Sim*$_{\varepsilon 2}$	*Range_Sim*$_{\varepsilon 3}$	*KNN*
Value				
Parameter	$\varepsilon_2 > \varepsilon_1$	$\varepsilon_2 > \varepsilon_1$	$\varepsilon_3 > \varepsilon_1$	K=3

In current approaches, the following predicates are considered different and analyzed separately:

- P_1 and P_2: two range queries with different parameters (radius)
- P_1 and P_3: two range queries with different parameters and values
- P_3 and P_4: two different operators

However, in multimedia applications, P_1 would also retrieve objects belonging to results of queries based on P_2 and P_3. Likewise, P_4 may return a subset of P_3's results. Thus, we can say that P_2 and P_3 infer P_1 (denoted by $P_1 \rightarrow P_2,P_3$), and consider only the results returned by P_2 / P_3 , thus eliminating P_1.

[1] More details about multimedia operators will be given later.

It is important to notice that ignoring such implications between predicates can lead, in multimedia applications, to higher computation costs when creating fragments, bigger fragments which is very restrictive for multimedia storage, migration, and retrieval, as well as data duplication on several sites. In [2, 11], the authors have only highlighted the implication issue importance, but have not well detailed nor identified the various kinds of implications. These issues will be tackled in following paragraphs.

4 Multimedia Primary Horizontal Fragmentation

In this section, we start by introducing some concepts and definitions necessary to tackle multimedia primary horizontal fragmentation. We develop subsequently additional steps to be integrated in current approaches, allowing adequate multimedia data fragmentation processing.

4.1 Definitions

4.1.1 Multimedia Object

A multimedia object is described by a set of attributes, related to a set of meta-data. It can be formally depicted as a set of attribute (a_i) and value (v_i) doublets:
O {(a_1, v_1); (a_2, v_2), … , (a_n, v_n)}. Multimedia attributes and values can be *simple* (like color = “red”), *complex* (color histogram, texture, shape, etc.) or the raw data (BLOB files) of multimedia objects.

4.1.2 Multimedia Type

A multimedia type allocates a set of attributes used to describe multimedia objects corresponding to that type: $T(a_1, a_2, a_3, \dots, a_n)$.We consider that two objects, described by the same attributes, are of the same type.

4.1.3 Multimedia Query

A multimedia query is written as follows [2, 9]:
q = {(Target clause), (Range clause), (Qualification clause)}, where:

- ***Target clause*:** contains multimedia attributes returned by the query
- ***Range clause*:** gathers the entities (tables/lasses) accessed by the query, to which belong *target clause* and *qualification clause* attributes
- ***Qualification clause*:** is the query restriction condition, a Boolean combination of predicates, linked by logical connectives ∧, ν, ¬

4.1.4 Multimedia Operators and Predicates

As mentioned before, multimedia information introduces new types of data and new operators and predicates. In the following, we explain multimedia operators and predicates related to low-level features. Note that semantic similarity operators are out of this paper’s scope and will be detailed in future studies.

4.1.4.1 Multimedia Operators

In multimedia databases, objects are widely described using vector spaces with numeric attributes, such as shape or color descriptors. Thus, in order to retrieve multimedia data, dedicated similarity queries are used, involving *range queries* and/or *k-nearest neighborhood* operators. Formal definitions are given thereafter.

4.1.4.1.1 Multimedia Range Query Operator

A range query operator $\bar{\theta}$ returns the set of objects V_j of an object value V_i located within a certain range ε from V_i using a distance function *D* (cfr. *Figure 1*). It can be formally written as:

$$\text{Range Query}(V_i, \bar{\theta}, \varepsilon) = N_{\bar{\theta}}^{\varepsilon}(V_i) = \{V_j / D(V_i, V_j) \le \varepsilon / \varepsilon \in \mathbb{R} \qquad \textbf{(1)}$$

The function *D* can be the classic Euclidean distance, a weighted Euclidean distance, a quadratic form distance, etc.

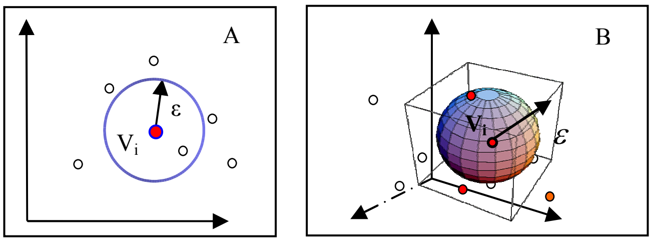


Fig. 1. Visualizations of a range query operator $\bar{\theta}$

A range query operator $\bar{\theta}$ has the following interesting properties, useful for optimizing the computation process:

- $N_{\bar{\theta}}^{\varepsilon_i}(V_i) \subseteq N_{\bar{\theta}}^{\varepsilon_j}(V_i)$ *if* $\varepsilon_i \le \varepsilon_j$
- if $N_{\bar{\theta}}^{\varepsilon_i}(V_i) \subseteq N_{\bar{\theta}}^{\varepsilon_j}(V_j)$ *and* $N_{\bar{\theta}}^{\varepsilon_j}(V_j) \subseteq N_{\bar{\theta}}^{\varepsilon_l}(V_l) \rightarrow N_{\bar{\theta}}^{\varepsilon_i}(V_i) \subseteq N_{\bar{\theta}}^{\varepsilon_l}(V_l)\ \forall \varepsilon_i, \varepsilon_j, \varepsilon_l$

4.1.4.1.2 Multimedia KNN Operator

A K-Nearest Neighborhood (KNN) operator $\vec{\theta}$ returns the set of K neighbors of an object value V_i located into either a ranged or unlimited domain space, using a distance *D* [3, 20]. It could be formally written as follows:

$$\text{KNN}(V_i, \vec{\theta}, k)_\varepsilon = N_{\vec{\theta}}^{k}(V_i)_\varepsilon = \left\{V_{j=1..k} \ / \ D(V_i, V_j) \le D(V_i, V)\right\}$$
$$\forall\ V \notin N_{\vec{\theta}}^{k}(V_i), \text{ where } k \in \mathbb{N} \text{ and } \text{Max}(D(V_i, V)) \le \varepsilon \ / \ \varepsilon \in \mathbb{R}^* \cup \{\perp\} \qquad \textbf{(2)}$$
$$\text{If } \varepsilon = \perp \text{, the domain space is unlimited}$$

As for range query operators, a KNN operator can be observed as a visual object in function of values dimensions. *Fig. 2* shows a ranged 2D KNN operator with k=3.

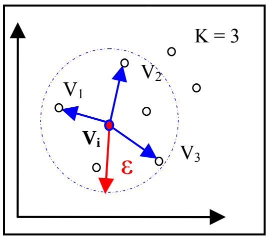


Fig. 2. Visualizations of a ranged 2D KNN operator $\vec{\theta}$

A KNN operator $\vec{\theta}$ has the following properties:

- $N^{k_i}_{\vec{\theta}}(V_i) \subseteq N^{k_j}_{\vec{\theta}}(V_i)$ *if* $k_i \leq k_j$
- if $N^{k_i}_{\vec{\theta}}(V_i) \subseteq N^{k_j}_{\vec{\theta}}(V_j)$ *and* $N^{k_j}_{\vec{\theta}}(V_j) \subseteq N^{k_l}_{\vec{\theta}}(V_l) \rightarrow N^{k_i}_{\vec{\theta}}(V_i) \subseteq N^{k_l}_{\vec{\theta}}(V_l)\ \forall k_i, k_j, k_l$

4.1.4.2 Multimedia Predicates

A multimedia predicate $\hat{P}$ is defined as follows:

$$\hat{P}_i = (A_i\ \theta_m\ V_i)$$

Where:

- A_i is a multimedia attribute or object
- V_i is a value in the domain of A_i or a multimedia object

$\theta_m = \theta_t \cup \{\bar{\theta}, \vec{\theta}\}$ *where* θ_t *is* a traditional operator such as a comparison operator (=, <, ≤, >, ≥, ≠), or a set operator (contained-in, set-equality, …), etc.

4.2 Steps for Multimedia Data Primary Horizontal Fragmentation

Before applying current fragmentation approaches, several steps should be executed in order to support and provide relevant multimedia data fragmentation. We suggest integrating the steps detailed below.

```
Multimedia_fragmentation_pre-processing ()

        Begin
                Multimedia_Types_Classification()
                For each multimedia Type
                        Predicates_Grouping()
                        Multimedia_Predicates_implication()
                EndFor
        End
```

4.2.1 Classification of Multimedia Objects

By applying existing horizontal fragmentation algorithms to a multimedia database, we attain non consistent horizontal fragmentation criteria (minterms). Suppose that *Camera Position*, *Audio Frequency* and *Dominant Color* are three multimedia attributes describing Video, Audio and Image objects respectively. The following Boolean expression: *CameraPosition = "North West"Λ AudioFrequency = "6 KHz" Λ DominantColor = ((10; 10; 10), RGB)* is a non consistent minterm, specifying criteria on "heterogeneous" attributes describing multimedia objects of different types, therefore producing an empty horizontal fragment.

In order to attain coherent minterms, we need to gather related objects together. As mentioned before, we assume that multimedia objects having the same attributes are considered of the same type. The algorithm provided below is used for classifying objects, according to their corresponding types.

Multimedia_Types_Classification ()

```
Input : MM        // multimedia objects
Output : T_M      //set of multimedia types corresponding to objects in MM

Begin
    For each Mo_i ∈ MM
        If Mo_i.A ≠ all T_i.A                  // Adding a new type corresponding to the object Mo_i
            New T_n+1 / T_n+1.A = Mo_i.A       // if the type isn't considered yet in MM
            T_n+1 = T_n+1 U Mo
        Else
            T_i = T_i U Mo_i / Mo_i.A = T_i.A  // Adding the object Mo_i to its corresponding type
        EndIf                                  // if the type is already identified
    Endfor
End
```

4.2.2 Predicate Grouping

It is also important to gather predicates into groups on the basis of operators. Using the algorithm below, two predicate groups are identified: multimedia and traditional. This separation will allow defining appropriate methods for multimedia implication:

$$P_i \xrightarrow[\theta_m]{} P_j \Leftrightarrow \begin{bmatrix} \widehat{P_i} \xrightarrow[\theta]{} \widehat{P_j} \\ P_i \xrightarrow[\theta_t]{} P_j \end{bmatrix} \qquad \xrightarrow[\theta]{} \text{ denotes a multimedia similarity implication} \qquad \xrightarrow[\theta_t]{} \text{ denotes a traditional implication}$$

Recall that traditional implication is out of this paper's scope.

Predicates_grouping ()

```
Input:  Q          //set of all user queries
        T_i        //a multimedia type
Output: P_j^i      //a query predicate defined on type T
        P̂_i        //set of multimedia predicates applied on T
        P_i        //set of traditional predicates applied on T
Begin
    For each query Q_i ∈ Q
        For each P_j^i ∈ Q_i
            If ( P_j^i ∈ P̂ ) then
                P̂_i = P̂_i ∪ P_j^i
            Else
                P_i = P_i ∪ P_j^i
            Endif
        EndFor
    EndFor
End
```

4.2.3 Multimedia Predicates Implication

Finding inference or implication between predicates is crucial to cutback the number of predicates involved in the fragmentation process [4, 11] (a large number of unnecessary fragments would notionally achieve low system performance). When a predicate P_i implies a predicate P_j (denoted by $P_i \rightarrow P_j$), P_i can be removed from the minterm fragment to which it belongs and replaced by P_j. Predicate implication is taken into consideration in traditional algorithms, mainly in Com_Min [12] and Make_Partition algorithms [10]. In the following, we detail the rules that can be used to determine implication between low-level feature-based predicates, by using both: range query and KNN methods.

4.2.3.1 Range Query Predicates Implication

Two range query predicates $\overline{P_i}$ *and* $\overline{P_j}$ are in implication if:

$$\overline{P_j} \rightarrow \overline{P_i} \Leftrightarrow \{0 \le D(V_i, V_j) \le \varepsilon_i - \varepsilon_j\}$$

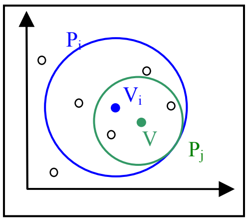


Fig. 3. 2D Range Query Predicates Implication

However, if $\varepsilon_i = \varepsilon_j$ and $D(V_i, V_j) \neq 0$ or if $\varepsilon_i - \varepsilon_j < D(V_i, V_j) \le \varepsilon_i + \varepsilon_j$, then there is an intersection between $\overline{P_i}$ *and* $\overline{P_j}$. Therefore, $\overline{P_i}$ *and* $\overline{P_j}$ cannot be associated via implication.

4.2.3.2 KNN Predicates Implication

The KNN implications for ranged or unlimited domain space are identical and can only be computed as follows:

$$\vec{P_j} \rightarrow \vec{P_i} \Leftrightarrow \{V_i = V_j \ \text{and} \ k_i \ge k_j\}$$

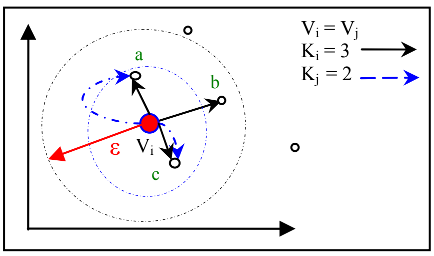


Fig. 4. KNN Predicates implication with identical values

Note that two KNN predicates $\vec{P_i}$ *and* $\vec{P_j}$ identified within two limited ranges ε_i and ε_j are not in implication (like for range queries) if:

$$\left\{0 < D(V_i, V_j) \le \varepsilon_i - \varepsilon_j \text{ where } \varepsilon_i \text{ and } \varepsilon_j \in [0,1]\right\}$$

4.2.3.3 Multimedia Predicates Implication

Using the same reasoning, we consider that two multimedia predicates $\widehat{P_i}$ *and* $\widehat{P_j}$ are in implication if:

$$\widehat{P_j} \rightarrow \widehat{P_i} \Leftrightarrow \left\{\begin{array}{l} 0 \le D(V_i, V_j) \le \varepsilon_i - \varepsilon_j \text{ and } \left(\widehat{P_i} \vee \widehat{P_j} \in \left\{\overline{P}\right\} \text{ and } (\vec{\varepsilon} < \overline{\varepsilon})\right) \\ \text{OR} \\ V_i = V_j \text{ and } k_i \ge k_j \end{array}\right\}$$

The first condition allows computing the implication between either two range query predicates or a range query predicate and a ranged KNN predicate. $\vec{\varepsilon}$ is used to designate the range of KNN predicate, and $\overline{\varepsilon}$ to designate the radius of the range predicate. The second condition highlights KNN predicates implication.

The following algorithm generates sets of multimedia predicate implications, IS_i, corresponding to each multimedia type T_i. Note that every set element consists of a doublet of predicates (P_i, P_j), meaning that P_i implies P_j.

Multimedia_Predicates_Implication ()

```
Input:    P̂i            //set of M multimedia predicates applied on a multimedia type T
Output:   ISi           //set of multimedia predicates implications applied on a type Ti
Variable: P^i_j         //a query predicate defined on type T
Begin
    For each P^i_j ∈ P̂i
        If j≤M-1 then
            For each P^i_j+1 ∈ P̂i
                If (Aj=Aj+1) then                                          //same attribute
                    If(P^i_j.operator = θ̄ and(P^i_j+1.operator = θ̄ or P^i_j+1.operator = θ⃗ )) then
                        If (εj > εj+1 ) then                               // Rj Rj+1 , Rj Kj+1
                            If 0 ≤ D(Vj,Vj+1) ≤ εj - εj+1 then             //P^i_j+1 → P^i_j
                                ISi=ISi ∪ (P^i_j+1,P^i_j)
                            Endif
                        Elseif (εj+1 > εj and P^i_j+1.operator = θ̄ ) then  // Rj Rj+1
                            If 0 ≤ D(Vj+1,Vj) ≤ εj+1 - εj then             //P^i_j → P^i_j+1
                                ISi=ISi ∪ (P^i_j,P^i_j+1)
                            Endif
                        Endif
                    Elseif (P^i_j.operator = θ⃗ and P^i_j+1.operator = θ⃗ ) then   // Kj Kj+1
                    If D(Vi,Vj) = 0 or Vi=Vj then
                        If (kj ≥ kj+1) then
```

```
                    IS_i=IS_i ∪ (P^i_{j+1}, P^i_j)
                Elseif (k_{j+1} ≥ k_j) then
                    IS_i=IS_i ∪ (P^i_j, P^i_{j+1})
                Endif
            Endif
        Elseif(P^i_j.operator = θ⃗ and P^i_{j+1}.operator = θ̄ ) then             // K_j R_{j+1}
            If (ε_{j+1} > ε_j) then
                If 0 ≤ D (V_{j+1}, V_j) ≤ ε_{j+1} − ε_j then                    // P^i_j → P^i_{j+1}
                    IS_i=IS_i ∪ (P^i_j, P^i_{j+1})
                Endif
            Endif
        Endif
      Endfor
    Endif
  Endfor
  IS_i = Optimize(IS_i)
End
```

```
Optimize(IS_i)
 Input: IS_i                      // set of multimedia predicates implications applied on a type T
  Begin
    For each (P^i_j, P^i_k) ∈ IS_i
       For each (P^i_k, P^i_l) ∈ IS
          If (P^i_j → P^i_k and P^i_k → P^i_l) then
             IS_i=IS_i ∪ (P^i_j, P^i_l)
          Endif
       EndFor
    EndFor
 End
```

4.2.4 Algorithm Complexity

The complexity calculations are carried out below on the basis of the worst case analysis. Suppose n_f represents the largest number of possible fragments, n_o represents the largest number of multimedia objects in a type or a fragment, n_q the largest number of user queries, n_t the largest number of types, n_p the largest number of multimedia predicates, n_i the largest cardinality of the sets IS_i, n_v the largest feature vector dimension involved. Our fragmentation pre-processing algorithm is of time complexity of $O(n_t \times (n_o + n_q \times n_p + n_v \times n_p^2 + n_i^2))$, which simplifies to $O(n_t \times (n_v \times n_p^2))$. Note that the polynomial (quadratic) nature of our features implication computation algorithm ($O(n_v \times n_p^2)$) dominates the complexity formulae and is experimentally demonstrated in our simulation prototype.

4.2.5 Computation Example

In the following, multimedia predicates (range query and KNN) will be illustrated in the same manner for the sake of simplicity:

$\overline{P}$ = A Similar(ε) V and $\vec{P}$ = A Similar(k, ε) V where:

– *A* is a multimedia attribute. In the present example, *A* stands for *Dominant Color* : *DC*
– *Similar* represents $\overline{\theta}$, the range similarity operator, when the number between brackets ε denotes a real value such as $0.0 \leq \varepsilon \leq 1.0$; ε designating the similarity range
– *Similar* stands for $\vec{\theta}$, the KNN operator, when the number between brackets k denotes an integer value ; k representing the number of neighboring objects to be returned by the KNN predicate within a range ε

Figure 5 shows three images a, b and c characterized by their feature vector values V_a, V_b and V_c respectively ; V designating, for each image, its *Dominant Color* feature in RGB color space (vector dimension = 3).

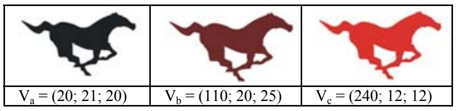


Fig. 5. Sample images

We also consider the following two range query predicates:
– $\mathbf{P_1}$**:** DC Similar(ε_1) V_1 and $\mathbf{P_2}$**:** DC Similar(ε_2) V_2 (*DC: Dominant Color*) where $V_1 = (22; 22; 22)$, $V_2 = (90; 10; 10)$, $\varepsilon_1 = 0.6$, and $\varepsilon_2 = 0.2$

Please note that in our similarity computations, we used the following weighted Euclidean distance function:

$$Dist(X,Y) = \frac{\sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}}{\sum_{i=1}^{N}(x_i + y_i)} \in [0,1]$$

N = Max (dim(X), dim(Y)), dim(X) and dim(Y) being the dimensions of vectors X and Y respectively.

Following our multimedia implication computation rules, predicate p_2 implies predicate p_1 ($0 \leq Dist(V_1, V_2) \leq \varepsilon_1 - \varepsilon_2$) where:
– $Dist(V_1, V_2) = ((22\text{-}90)^2 + (22\text{-}10)^2 + (22\text{-}10)^2)^{1/2} \ / \ (22 + 90 + 22 + 10 + 22 + 10) = 0.397$
– and $\varepsilon_1 - \varepsilon_2 = 0.6 - 0.2 = 0.4$

A query utilizing predicate P_1 would return still regions a and b
– $Dist(V_1, V_a) = 0.024$ ($< \varepsilon_1$, returned object)
– $Dist(V_1, V_b) = 0.399$ ($< \varepsilon_1$, returned object)
– $Dist(V_1, V_c) = 0.662$ ($> \varepsilon_1$)

Whereas a query invoking predicate P_2 would return still region b
– $Dist(V_2, V_a) = 0.417$ ($> \varepsilon_2$)
– $Dist(V_2, V_b) = 0.102$ ($< \varepsilon_2$, returned object)
– $Dist(V_2, V_c) = 0.401$ ($> \varepsilon_2$)

One can clearly realize that the set of multimedia objects returned by P_1 ({a, b}) includes those returned by of P_2 ({b}). If taken into account, such implications would reduce fragment creation computation cost, fragment size and multimedia data duplication on multiple sites.

5 Prototype

To validate our approach, we have implemented a C# prototype called “Multimedia Implication Identifier” encompassing:

- A relational database, storing multimedia objects via Oracle 9i DBMS, described following the multimedia meta-model M² (MPEG-7 compatible) developed by Chalhoub *et al.* in [4].
- A set of interfaces allowing users to formulate simple and complex multimedia queries, providing the ability to select multimedia information.
- Containers for storing user queries, enabling, via specific processes, the computation of query access frequencies which are basically used in the predicate affinity calculations.
- Specific containers undertaking the storage of predicates, utilized by dedicated procedures to calculate predicate implications.

The prototype accepts, as input, multimedia queries. Automatic processes subsequently calculate query access frequencies, identify corresponding predicates, and compute for each multimedia type (represented by a table) its Predicate Usage Matrix (PUM)[1] and its Predicate Affinity Matrix (PAM)[2] used to measure the affinity between predicates, the PAM taking into account our predicate implication steps.

Note that we chose to present multimedia implications in PAM matrixes, proposed by [15, 4], for the sake of clearness (PAMs being suitable structures for displaying predicate implications). Nevertheless, our algorithm is generic in the sense that it could be equally used with other primary horizontal fragmentation approaches, Com_Min [16] in particular.

5. 1. Simulation example

Among the various tests that were conducted, we present a simple simulation example comparing predicate affinities (PAM) obtained with and without the inclusion of our multimedia physical implication rules. In the following example, multimedia type “Still Region”, designating motionless images, is selected for PUM and PAM calculations. Let $Q = \{q_{i\ =\ 0\ to\ 5}\}$ be a set of user queries defined on “Still Region” Type. Recall that we represent queries following paragraph 4.1.3.

q₀: { (MO); (StillRegion); (ObNature = "vehicule" ∧
DC Similar(0.3) ((12; 10; 13), (14; 15; 16), (20; 20; 20))) }
q_1: { (MO); (StillRegion); (ObNature = "vehicule" ∧ ObColor = "red" ∧
DC Similar(0.2) ((12; 10; 13), (14; 15; 16), (20; 20; 20))) }
q_2: { (MO); (StillRegion); (ObNature = "truck" ∧ ObColor = "red" ∧
DC Similar(0.1) ((9; 8; 7), (7; 8; 7), (10; 11; 10))) }
q_3: { (MO); (StillRegion); (ObNature = "vehicule" ∧
DC Similar(3) ((12; 10; 13), (14; 15; 16), (20; 20; 20))) }
q_4: { (MO); (StillRegion); (ObNature = "vehicule" ∧ ObColor = "red" ∧
DC Similar(1) ((12; 10; 13), (14; 15; 16), (20; 20; 20))) }
q_5: { (MO); (StillRegion); (ObNature = "truck" ∧ ObColor = "red" ∧
DC Similar(1) ((9; 8; 7), (7; 8; 7), (10; 11; 10))) }

[1] It contains the predicates used by each query as well as query access frequencies and is subsequently used as input to the PHF process adopted by [11, 2]

[2] Following [15, 4], the PAM is a square and symmetric matrix where each value aff(P_i, P_j) can be numerical or non numerical. Numerical affinity represents the sum of the frequencies of queries which access simultaneously P_i and P_j. Non numerical affinity underlines the implication relation between predicates P_i and P_j

Let $P = \{P_i, i = 0 \text{ to } 8\}$ be the set of predicates used by Q.

P_0: ObNature = "vehicule"
P_1: DC Similar(0.3) ((12; 10; 13), (14; 15; 16), (20; 20; 20))
P_2: ObColor = "red"
P_3: DC Similar(0.2) ((12; 10; 13), (14; 15; 16), (20; 20; 20))
P_4: ObNature = "truck"
P_5: DC Similar(0.1) ((9; 8; 7), (7; 8; 7), (10; 11; 10))
P_6: DC Similar(3) ((12; 10; 13), (14; 15; 16), (20; 20; 20))
P_7: DC Similar(1) ((12; 10; 13), (14; 15; 16), (20; 20; 20))
P_8: DC Similar(1) ((9; 8; 7), (7; 8; 7), (10; 11; 10))

P contains traditional predicates (P_0, P_2) as well as multimedia predicates (P_1, P_3, P_4, P_5, P_6, P_7, P_8). Note P_1, P_3 and P_5 are range query predicates (the number between brackets being a real value – similarity range ε), while P_6, P_7 and P_8 are KNN predicates (the number between brackets being an integer value – number of objects k to be returned by the predicate). Also note that *DC* represents a composite *Dominant Color* feature vector stating the three consecutive dominant colors in an image, in RGB color space. For example, DC_1 of predicate p_1 underlines dominant colors C(12; 10; 13), C'(14; 15; 16) and C''(20; 20; 20).
By reading the updated PAM, one can clearly point out the multimedia implication rules defined in the paper:

- Predicate P_3 ($\varepsilon_3 = 0.2$, V_3 = ((12; 10; 13), (14; 15; 16), (20; 20; 20))) implies P_1 ($\varepsilon_1 = 0.3$, V_1 = ((12; 10; 13), (14; 15; 16), (20; 20; 20))) having:
  - $V_1 = V_3$ and $\varepsilon_1 > \varepsilon_3$
- Predicate P_5 ($\varepsilon_5 = 0.1$ "max", V_5 = ((9; 8; 7), (7; 8; 7), (10; 11; 10))) implies P_1 ($\varepsilon_1 = 0.3$, V_1 = ((12; 10; 13), (14; 15; 16), (20; 20; 20))) having:
  - $\varepsilon_1 > \varepsilon_5$, $dist(V_1,V_5) \leq \varepsilon_1 - \varepsilon_5$
- No implication can be identified between predicates P_3 and P_5 having:
  - $dist(V_3,V_5) > \varepsilon_3 - \varepsilon_5$ (similarity circle intersection/exclusion)
- Predicate P_7 ($k_7 = 1$, V_7 = ((12; 10; 13), (14; 15; 16), (20; 20; 20))) implies predicate P_6 ($k_6 = 3$, V_6 = ((12; 10; 13), (14; 15; 16), (20; 20; 20))) having:
  - $V_6 = V_7$ and $k_6 > k_7$
- No implication can be identified between P_6 (orP_7) and P_8, having:
  - $V_8 \neq V_6$ (correspondingly V_7)

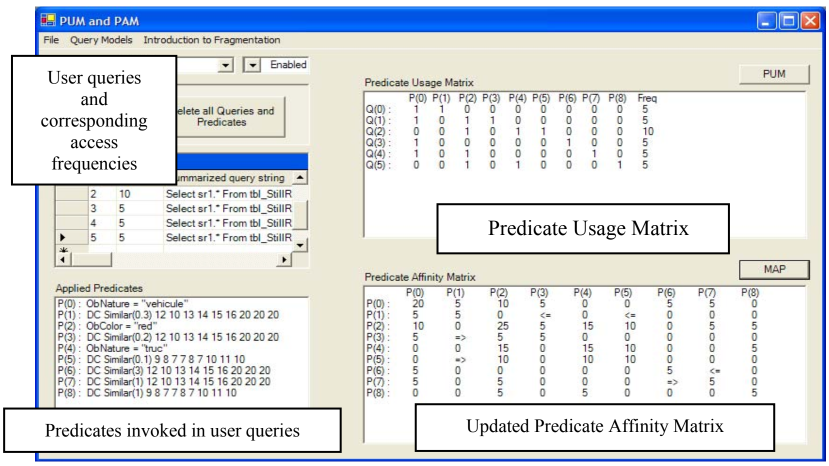


Fig. 6. Updated Predicate Affinity Matrix.

Disregarding our multimedia implication rules would yield, in the present example, a PAM with only numerical affinities.
The PUM and uPAM make up the inputs to the NHP primary horizontal partitioning algorithm [11, 2], not being implemented yet in our prototype.

5.2 Timing Analysis

We have shown that the complexity of our physical similarity implication simplifies to $O(n_v \times n_p^2)$. We verified the formula experimentally, the timing results being presented in *Fig. 7*.

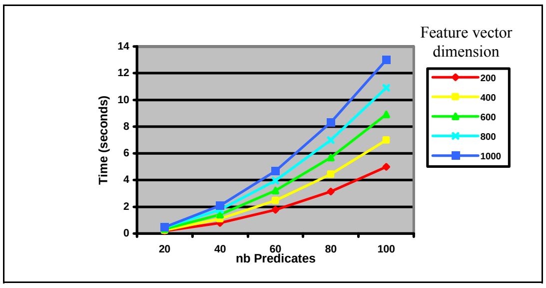


Fig. 7. Timing results

The experiment was carried out on a Pentium 4 PC (2.8 Ghz CPU, 798 Mhz bus, 512 MB RAM). One can see that the time to compute similarity implications grows in a polynomial (quadratic) fashion with the number of predicates involved. Our experiments also show that feature vector dimension affects time complexity, owing to predicate distance computations (weighted Euclidian distance).

6 Conclusion and Future Work

In this paper, we proposed an approach for the Primary Horizontal Fragmentation of multimedia databases, by extending existing fragmentation methods. Following the definition of a multimedia type, we identified the need to classify multimedia objects corresponding to the same type, in order to achieve consistent horizontal fragmentation criteria. The "Type Fragmentation" phase could be then followed by the PHF of each generated type. The original idea of emerging new multimedia operators allowed the adaptation of existing fragmentation procedures to partition multimedia data. We concentrated our efforts on the primary horizontal fragmentation of unstructured multimedia data, emphasizing the impact of multimedia predicate implications in optimizing multimedia fragments.
Future directions include the introduction of semantic-based multimedia predicates. Our future goals also incorporate generating a multimedia conceptual schema, including the derived horizontal fragmentation process, and optimizing, if possible, the used fragmentation methods (semantic implication is yet to be developed). Likewise, multimedia vertical fragmentation and XML fragmentation will be talked in upcoming studies.

References

1. Baiao F, Mattoso M., A Mixed Fragmentation Algorithm for Distributed Object Oriented Databases. *9th Inter. Conf. on Computing Information*, Canada, 1998
2. Belatreche L, Karlapalem K, Simonet A., Horizontal class partitioning in object-oriented databases. *8th Inter. Conf. on Database and Expert Systems Applications* (DEXA' 97), Toulouse, September 1997
3. Braunmuller B, Ester M, Kreigel H. P., Sander J., Efficiently Supporting Multiple Similarity Queries for Mining in Metric Databases. *Proc. of the 16th Inter. Conf. on Data Engineering*, p.256, 2000
4. Chalhoub G, Saad S, Chbeir R, Yetongon K., A Multimedia Meta-Database Model for Distributed Multimedia DBMS. *WSEAS*, 2004
5. Chinchwadkar G.S., Goh A., An Overview of Vertical Partitioning in Object Oriented Databases. *The Computer Journal,* Vol. 42, No. 1, 1999
6. Ezeife C.I., Barker K., A Comprehensive Approach to Horizontal Class Fragmentation in a Distributed Object Based System. *Inter. Journal of Distributed and Parallel Databases*, 1, 1995. Kluwer Academic Publishers
7. Ezeife C.I., Barker K., Distributed Object Based Design: Vertical Fragmentation of classes. *Journal of Distributed and Parallel DB Systems*, 6(4): 327-360, 1998
8. Kosch H., Distributed Multimedia Database Technologies Supported by MPEG-7 and MPEG-21, Auerbach Publications, 280 p., 2004
9. Navathe S.B, Ceri S, Wiederhold G, Dou J., Vertical Partitioning Algorithms for Database Design. *ACM Transactions on Database Systems*, 9, 680-710, 1984
10. Navathe B, RA M., Vertical Partitioning for Database Design: a Graphical Algorithm. *1989 ACM SIGMOND Conf.*, Portland, p. 440-450, 1989
11. Navathe S.B, Karlapalem K, Ra M., A Mixed Partitioning Methodology for Initial Distributed Database Design. *Journal of Computer and Software Engineering*, 3(4): 395-426, 1995
12. Ozsu M.T, Valduriez P., Principals of Distributed Database Systems. Prentice Hall, Prentice Hall, 1991
13. Gertz M, Bremer J.M., Distributed XML Repositories: To-Down Design and Transparent Query Processing. Department of CS, University of California, 2004
14. Sub C., An approach to the model-based fragmentation and relational storage of XML-documents. Grundlagen von Datenbanken 2001:98-102
15. Sub C. et al., Data Modeling and Relational Storage of xml-based Teachware. GI Jahrestatung (1) 2001:378-387
16. Grosky W. I., Managing Multimedia Information in Database Systems, Communications of the ACM, 1997, Vol. 40, No. 12, pp. 72-80
17. Synchronized Multimedia Working Group, www.w3.org/tr/rec-smil, 02-02-2006
18. SVG Working Group: www.w3.org/tr/svg., 02-12-2006
19. MovingPictureExperts Group: http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm, 02-27-2005
20. Bernhard Braunmuller et al., Efficiently Supporting Multiple Similarity Queries for Mining in Metric Databases, *IEEE Trans. on Knowledge and Data Engineering,* v.13 n.1, p.79-95, January 2001