Towards Efficient Horizontal Multimedia Database Fragmentation using Semantic-based Predicates Implication

Fekade Getahun[1], Joe Tekli[2], Solomon Atnafu[1], Richard Chbeir[3]

[1] Department of Computer Science, Faculty of Informatics
Addis Ababa University, 1176 Addis Ababa, Ethiopia

[2] LE2I Laboratory UMR-CNRS, University of Bourgogne
21078 Dijon Cedex France

{fekadeg, satnafu}@cs.aau.edu.et,
{joe.tekli, richard.chbeir}@u-bourgogne.fr

Abstract. *Partitioning techniques are traditionally used in distributed system design to reduce accesses to irrelevant information by grouping data frequently accessed together in specific fragments. Here, we address the primary horizontal fragmentation of textually annotated multimedia data. In this study, we discuss the issue of identifying semantic implications between textual-based multimedia predicates, as a crucial phase in the efficient partitioning of multimedia data. Our proposal integrates knowledge bases as a framework for assessing the semantic relatedness between predicate values and operators. We developed a prototype implementing the various aspects of multimedia semantic predicates implications. Experimental results show that the proposed method is polynomial in the number of user predicates as well as the sizes of the knowledge bases being employed. Real-world multimedia fragmentation tests are ongoing.*

1. Introduction

Video and audio-on-demand, video conferencing, distance e-Learning and cartography are only few examples of multimedia applications emerging on the web. In such a distributed environment, many technical problems need to be solved in order to obtain a full-fledged distributed multimedia database system. These problems concern all layers of the multimedia system, in particular the storage and retrieval layers.

Traditionally, partitioning techniques are used in distributed system design to improve data storage and retrieval efficiency. Three main fragmentation techniques have been defined for relational databases: horizontal, vertical and hybrid. More recently, some researchers have extended these techniques for partitioning object oriented databases. In essence, fragmentation consists of dividing the database objects and/or entities into fragments, on the basis of common queries accesses, in order to distribute them over several distant sites. The fragmentation enhances system performance [Ezeife and Barker, 1995] by:

- Reducing the amount of irrelevant data accessed by applications, (because applications usually access portions of entities and objects),
- Allowing parallel execution of a single query, dividing it into a set of sub-queries that operate on segments of an entity/class,
- Reducing the quantity of data transferred when migration is required,
- Decreasing data update cost and storage space.

Several continuing studies are aimed at building distributed MultiMedia DataBase Management Systems MMDBMS [Kosch 2004]. Nevertheless, most existing systems lack a formal framework to provide full-fledged multimedia operations. In particular, multimedia fragmentation remains 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 integration of metadata (consisting of semantic descriptors such as event, location, which person appears in a picture, etc., and/or low-level descriptors such as color, texture, shape, etc.) to describe the multimedia content.

In this paper, we address primary horizontal fragmentation (cf. Section 2) in distributed multimedia databases. We particularly address semantic-based predicates implication required in current fragmentation algorithms, such as *Make_Partition* and *Com_Min* [Oszu and Valduriez 1991], [Navathe *et al.* 1995], [Belatreche *et al.* 1997], in order to partition multimedia data efficiently. Since predicate implications are of crucial impact in traditional fragmentation techniques, we believe that the identification of semantic implications between multimedia predicates will improve the multimedia fragmentation process (as we will show in the motivation section). In this study, we introduce a set of algorithms for identifying semantic implications between predicate values, predicate operators, and consequently multimedia semantic-based predicates. We also present our prototype with some preliminary experiments to test and evaluate our approach. The remainder of this paper is organized as follows. Section 2 reviews background in DB fragmentation. In Section 3, we present a motivation example. Section 4 defines the concepts to be used in our approach. In Section 5, we detail our semantic implication algorithms and their usage in multimedia fragmentation. Section 6 presents our prototype and timing analysis. Finally, Section 7 concludes.

2. Background and Related Work

Fragmentation techniques for distributed DB systems aim to achieve effective resource utilization and improved performance [Chinchwadkar and Goh 1999]. This is addressed by removing irrelevant data accessed by applications and by reducing data exchange among sites [Baiao and Mattoso 1998]. In this section, we briefly present traditional database fragmentation approaches, and focus on horizontal fragmentation algorithms. We also report recent approaches targeting XML as well as multimedia data fragmentation.

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 [Ozsu and Valduriez 1991]. 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 [Baiao and Mattoso 1998]. MF is a hybrid partitioning technique where horizontal and vertical fragmentations are simultaneously applied on an entity/class [Navathe *et al.* 1995].

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. In this paper, we only focus on PHF which is, to the best of our knowledge, has been addressed mainly by two main algorithms: *Com_Min* developed in [Oszu and Valduriez 1991] and *Make_Partition* Graphical Algorithm developed in [Navathe and Ra 1989] (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 [Belatreche *et al.* 1997] 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* minterms [Navathe *et al.*

1997] (the number of minterms 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 in [Ezeife and Barker 1995] using Com_Min [Oszu and Valduriez 1991], and the other developed in [Bellatreche *et al.* 1997] on the basis of Make_Partition [Navathe and Ra 1989]. The use of Com_Min or Make_Partition is the major difference between them.

Recent works have addressed XML fragmentation [Sub 2001], [Gertz and Bremer 2004] due to the various XML-oriented formats available on the web. The usage of XPaths and XML predicates forms the common basis of all these studies. Yet, XML fragmentation methods are very specific and hardly applicable to multimedia databases.

A recent address to address multimedia database fragmentation is provided in [Saad. *et al.* 2006]. The authors here discuss multimedia primary horizontal fragmentation and provide a partitioning strategy based on the low-level features of multimedia data (e.g. color, texture, shape, etc., represented as complex feature vectors). They particularly emphasize the importance of multimedia predicates implications in optimizing multimedia fragments.

3. Motivation

In order to fragment multimedia databases, several issues should be studied and extended. Multimedia queries contain new operators handling low-level and semantic features. These new operators should be considered when studying predicates and particularly predicate implications [Saad *et al.* 2006]. For example, let us consider the following predicates used to search for videos in the movie database IMDB[1].

Table 1. Semantic predicates

Predicate	Attribute	Operator	Value
P_1	*Keywords*	=	*"Football"*
P_2	*Keywords*	=	*"Tennis"*
P_3	*Keywords*	=	*"Sport"*
P_4	*Location*	=	*"Coliseum"*
P_5	*Location*	*Like %*	*"Rome"*

In current fragmentation approaches, these predicates are considered different and are analyzed separately. Nonetheless, a multimedia query consisting of P_1 and P_2 would retrieve movies belonging to the result of P_3, the value/concept *Sport* encompassing in its semantic meaning *Football* and *Tennis*. Thus, we can say that P_1 and P_2 imply P_3 (P_1, $P_2 \Rightarrow P_3$). Consequently, the fragmentation algorithm should only consider P_3, eliminating P_1 and P_2 while generating fragments. A similar case can also be identified with P_4 and P_5. The value/concept *Rome* covers in its semantic meaning *Coliseum*. However, the operator used in P_4 is not the same as that utilized in P_5, which raises the question of operator implication. Since the operator *Like %* covers in its results those of the operator *equal* (*Like %* returning results that are identical or similar to a given value, where *equal* returns only the results identical to a certain value), the results of P_5 would cover those returned by P_4. Hence, we can deduce that P_4 implies P_5 ($P_4 \Rightarrow P_5$). As a result, the fragmentation algorithm should only consider P_5, disregarding P_4. Note that ignoring such implications between predicates can lead, in multimedia applications, to higher computation costs when creating fragments, bigger fragments which are very restrictive for multimedia storage, migration, and retrieval, as well as data duplication on several sites [Saad *et al.* 2006].

In [Navathe *et al.* 1995], [Belatreche *et al.* 1997], the authors have highlighted the importance of implication, but have not detailed the issue. As mentioned before, the authors in [Saad *et al.* 2006] have only addressed implications between low-level multimedia predicates

---

[1] Available at http://www.imdb.com/

(based on complex feature vectors). In this study, we go beyond low-level features provided in [Saad *et al.* 2006] and present a complementary semantic implication approach (Figure 3).

4. Preliminaries

In the following, we define the major concepts used in our approach. We particularly detail the notions of *Knowledge Base* (KB) and *Neighborhood* (N) which will be subsequently utilized in identifying the implications between semantic predicates.

4.1. Basic Definitions

Definition 1 - Multimedia Object: is depicted as a set of attribute (a_i) and value (v_i) doublets: O $\{(a_1, v_1), (a_2, v_2), \dots, (a_n, v_n)\}$. Multimedia attributes and values can be *simple* (numeric or textual fields), *complex* (color histogram, texture, shape, etc.) or contain raw data (BLOB files) of multimedia objects. Note that in horizontal multimedia fragmentation, multimedia objects constitute the basic reference units (similarly to 'objects' in object oriented DB partitioning and 'tuples' in relational DB fragmentation).

Definition 2 - Multimedia Type: allocates a set of attributes used to describe multimedia objects corresponding to that type [Saad *et al.* 2006]. Two objects, described by the same attributes, are of the same type.

Definition 3 - Multimedia Query: is written as follows [Belatreche *et al.* 1997], [Saad *et al.* 2006]: *q = {(Target clause), (Range clause), (Qualification clause)}*

- ***Target clause*:** contains multimedia attributes returned by the query,
- ***Range clause*:** gathers the entities (tables/classes) 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 $\wedge$, $\vee$, $\neg$.

Definition 4 - Multimedia predicate: is defined as $P=(A\,\theta\,V)$, where:

- A is a multimedia attribute or object,
- V is a value (or a set of values) in the domain of A,
- θ is a low-level multimedia operator (*Range* and *KNN* operators), a comparison operator θ_c (=, <, ≤, >, ≥, ≠, *like*) or a set operator θ_s (*in* and θ_c*qualifier* where the quantifiers are: *any*, *some*, *all*).

4.2. Knowledge Base

In the fields of Natural Language Processing (NLP) and Information Retrieval (IR), knowledge bases (thesauri, taxonomies and/or ontologies) provide a framework for organizing entities (words/expressions [Richardson and Smeaton 1995], [Lin 1998], generic concepts [Rodriguez and Egenhofer 2003] [Ehrig and Sure 2004], web pages [Maguitman 2005], etc.) into a semantic space. Subsequently, knowledge bases are utilized to compare/match the considered entities with respect to their corresponding similarity/relevance degrees with one another. In our approach, we employ knowledge bases as a reference for identifying semantic implications between predicates, which is not addressed in existing approaches. As shown in the motivating example, implication between semantic predicates relies on the implications between corresponding values and operators. Hence, two types of knowledge bases are used here: i) value-based: to represent the domain values commonly used in the application, and ii) operator-based: to organize operators used with semantic-based predicates. We will also give the semantic relations commonly used in the literature [Richardson and Smeaton 1995], [Lin 1998], [WordNet 2005], to organize entities and concepts in a KB. We detail them below.

4.2.1. Knowledge Base

In our study, a *Value Knowledge Base* (V_{KB}) is domain-oriented and comes down to a hierarchical taxonomy with a set of concepts representing groups of words/expressions (which we identify as *value concepts*), and a set of links connecting the values, representing semantic relations[1].

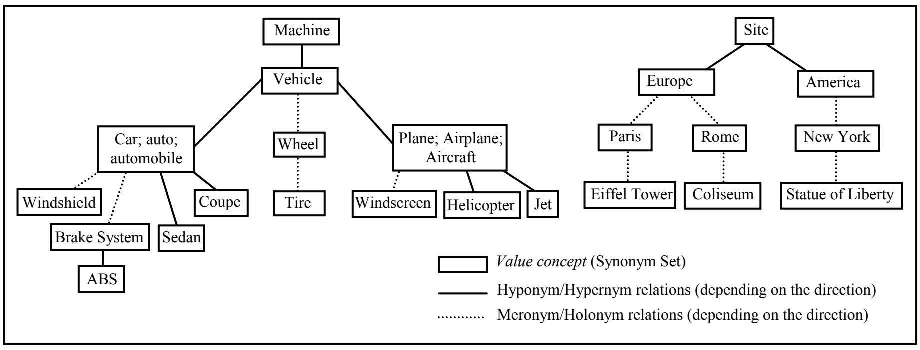


Figure 1. Sample value knowledge base with multiple root concepts

As in WordNet[2], we consider that a V_{KB} *concept* consists of a set of synonymous words/expressions such as *{car, auto, automobile}*. *Value concepts* are connected together via different semantic relations, which will be detailed subsequently. Formally, $V_{KB}=(V_c, E, R, f)$ where:

- V_c is the set of *value concepts* (synonym sets as in WordNet [Miller 1990]).
- E is the set of edges connecting the *value concepts*, where $E \subseteq V_c \times V_c$.
- R is the set of semantic relations, $R = \{\Omega, \prec, \succ, \ll, \gg\}$ (cf. Table 2), the synonymous words/expressions being integrated in the *value concepts*.
- f is a function designating the nature of edges in E, $f{:}E \rightarrow R$ (cf. Figure 1).

4.2.2. Operator Knowledge Base

As stated previously, operators should also be considered when studying the implication between semantic predicates. Therefore, an operator knowledge base of four descriptors $O_{KB}=(O_c, E, R, f)$ is also defined where:

- O_c is the set of *operator concepts*, consisting of mono-valued *comparison operators* θ_c (=, ≠, >, <[3], and *like*) as well as multi-valued *ones* θ_s (*in* and θ_c*qualifier* where the quantifiers are: *any*, *some*, *all*).
- E is the set of edges connecting the operators, where $E \subseteq O_c \times O_c$.
- R is the set of semantic relations, $R=\{\Omega, \prec, \succ, \ll, \gg\}$.
- f is a function designating the nature of edges in E, $f{:}E \rightarrow R$.

We designed the operator knowledge base O_{KB} as shown in Figure 2.

In the mono-valued operator taxonomy, we can particularly observe that the pattern matching operators *Like* and *Not Like* (considered as antonyms) make use of the parameters '_' and '%', to represent one and zero/multiple optional characters respectively. Hence, we represent this fact by a semantic *IsA* $\prec$ relation[4] following these operators, i.e. *Like_* $\prec$ *Like%*

[1] However, the building process of the value knowledge base is out of the scope of this paper.
[2] WordNet is an online lexical reference system (taxonomy), where nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing a lexical concept [Miller 1990], [WordNet 2005].
[3] ≥ and ≤ are considered as single operators put together using the Boolean operator OR.
[4] Relations will be detailed in the next subsection.

and *Not Like_* $\prec$ *Not Like%*. On the other hand, '<' and '>' implicitly denote the operator '≠' (commonly represented by < >), thus are considered as sub-operators of this later.

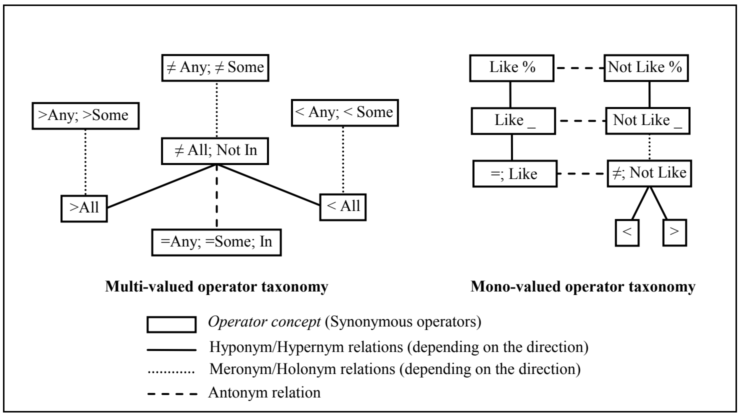


Figure 2. Our proposed operator knowledge base

In the multi-valued operator taxonomy, the *any* and *some* quantifiers are considered as synonyms, as well as the operators *≠All* and *Not In*, and *=Any* (or *Some*) and *In*. The *>All* and *<All* operators are considered as sub-operators of *≠All* (like mono-valued operators) and thus are linked to this later using *IsA* relations. In addition, the *>All* and *>Any* operators are linked together because if the condition is valid for all comparison values, it must be for any value inside the comparison set. Likewise for *<All* and *<Any*, and *≠All* and *≠Any*.

4.3. Semantic Relations

Hereunder, we develop the most popular semantic relations employed in the literature, which are included in the WordNet knowledge base [30, 31, 32]:

- *Synonym (≡)*: Two words/expressions (likewise for operators) are synonymous if they are semantically identical, that is if the substitution of one for the other does not change the initial semantic meaning.
- *Antonym (Ω)*: The antonym of an expression is its negation.
- *Hyponym* ($\prec$): It can also be identified as the *subordination* relation, and is generally known as the *Is Kind of* relation or simply *IsA*.
- *Hypernym* ($\succ$): It can also be identified as the *super-ordination* relation, and is generally known as the *Has Kind of* relation or simply *HasA*.
- *Meronym* ($\ll$): It can also be identified as the *part-whole* relation, and is generally known as *PartOf* (also *MemberOf*, *SubstanceOf*, *ComponentOf*, etc.).
- *Holonym* ($\gg$): It is basically the inverse of Meronym, and is generally identified as *HasPart* (also *HasMember*, *HasSubstance*, *HasComponent*, etc.).

Table 2 reviews the most frequently used semantic relations along with their properties [Richardson and Smeaton 1995] [Lin 1998], [WordNet 2005]. Note that the transitivity property is not only limited to semantic relations of the same type and could also exist between heterogeneous relations. For example:

- *Brake system* $\ll$ *car* and *car* ≡ *automobile* transitively infer *Brake system* $\ll$ *automobile.*
- *ABS* $\prec$ *Brake system* and *Brake system* $\ll$ *car* transitively infer *ABS* $\ll$ *car* (Figure 1).

Formally, let C_i, C_j and C_k be three concepts connected via semantic relations R_{ij} and R_{jk} in a given *KB*. Table 3 details the transitivity properties for all semantic relations defined in the previous subsections, identifying the resulting relation R_{ik} transitively connecting concepts C_i and C_k. The relevance of identifying transitivity between different semantic relations will be demonstrated when defining the neighborhood of a concept, subsequently recognizing the concept implications.

Table 2. Semantic relations

Relation \ Property	Symbol	Reflexive	Symmetric	Transitive
Synonym	$\equiv$	✓	✓	✓
Antonym	Ω	×	✓	×
Hyponym	$\prec$	✓	×	✓
Hypernym	$\succ$	✓	×	✓
Meronym	$\ll$	✓	×	✓
Holonym	$\gg$	✓	×	✓

Table 3. Transitivity between relations

R_{ij} \ R_{ik}	$\equiv$	Ω	$\prec$	$\succ$	$\ll$	$\gg$
$\equiv$	$\equiv$	Ω	$\prec$	$\succ$	$\ll$	$\gg$
Ω	Ω	$\equiv$	Ω	Ω	$\varnothing$	$\varnothing$
$\prec$	$\prec$	Ω	$\prec$	$\varnothing$	$\ll$	$\varnothing$
$\succ$	$\succ$	Ω	$\varnothing$	$\succ$	$\varnothing$	$\gg$
$\ll$	$\ll$	$\varnothing$	$\ll$	$\varnothing$	$\ll$	$\varnothing$
$\gg$	$\gg$	$\varnothing$	$\varnothing$	$\gg$	$\varnothing$	$\gg$

4.4. Neighborhood

In our approach, the neighborhood notion is used to compute the implication between values, operators, and consequently predicates. The implication neighborhood of a concept C_i is defined as the set of *concepts* $\{C_j\}$, in a given knowledge base *KB*, related with C_i via the synonym ($\equiv$), hyponym ($\prec$) and meronym ($\ll$) semantic relations, directly or via transitivity. It is formally defined as:

$$N^{R}_{KB}(C_i) = \left\{ C_j \ / \ C_i \ R \ C_j \ and \ R \in \{\equiv, \prec, \ll\} \right\} \quad \textbf{(1)}$$

When applying the neighborhood concept to some value concepts in Figure 1, we obtain the following implication neighborhood examples:

- $N^{\equiv}_{V_{KB}}(car) = \{car, auto, automobile\}$
- $N^{\prec}_{V_{KB}}(ABS) = \{ABS, brake\ system\}$
- $N^{\ll}_{V_{KB}}(tire) = \{tire, wheel, vehicle, machine\}$ (transitivity between $\ll$ and $\prec$)

Moreover, we define the global implication neighborhood of a concept to be the union of each implication neighborhood with respect to the synonym ($\equiv$), hyponym ($\prec$) and meronym ($\ll$) semantic relations:

$$\overline{N}_{KB}(C_i) = \cup N^{R}_{KB}(C_i) / R \in \{\equiv, \prec, \ll\} \quad \textbf{(2)}$$

Note hereunder the corresponding global neighborhoods of the same examples:

- $\overline{N_{V_{KB}}}(car) = N^{\equiv}_{V_{KB}}(car) \cup N^{\prec}_{V_{KB}}(car) \cup N^{\ll}_{V_{KB}}(car) = \{car, auto, automobile, vehicule, machine\}$

  $\overline{N_{V_{KB}}}(ABS) = \{ABS, brake\ system, car, auto, automobile, vehicle, machine\}$

Similarly, the implication neighborhood can be applied to operator concepts:

- The global neighborhood of the *Like* operator: $\overline{N_{O_{KB}}}(Like) = \{=, Like, Like\_, Like\%\}$.
- The global neighborhood of $\neq All$: $\overline{N_{O_{KB}}}(\neq All) = \{\neq All, Not\ In, \neq Any, \neq Some\}$.
- The global implication neighborhood of $>All$:

  $\overline{N_{O_{KB}}}(> All) = \{> All, > Any, > Some, \neq All, Not\ In, \neq Any, \neq Some\}$.

5. Semantic Implication Between Predicates

Finding implication between predicates is crucial to cutback the number of predicates involved in the fragmentation process [Navathe *et al.* 1995], [Belatreche *et al.* 1997] (a large number of unnecessary fragments would notionally achieve low system performance especially when using multimedia data). 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 can be replaced by P_j. In the following, we detail the rules that can be used to determine implication between semantic predicates. As mentioned earlier, the semantic implication between two predicates depends on the implications between their corresponding *values* and *operators*. Therefore, we develop value and operator implications before introducing our predicate implication algorithm. Our *Semantic Implication Algorithm* (*SPI*) is complementary to that developed in [Saad *et al.* 2006] and thus could be coupled with its overall process (cf. Figure 3) in order to enable relevant multimedia fragmentation. Due to the space limitation, value and operator neighborhood computation will not be detailed here since the main definitions have been already covered previously.

```
Multimedia_fragmentation_pre-processing ()     // Developed in [Saad et al. 2006] to the exception
                                               // of semantic implication.
Begin
   Multimedia_Types_Classification()           //Classifying multimedia objects according to their types
      For each multimedia Type
         Predicates_Grouping()                 //Grouping low-level and semantic predicates together
         Multimedia_Predicates_implication()   // Low-level predicates implications
         Semantic_Predicates_Implication()     // Contribution of our study.
      End For
End
```

Figure 3. Multimedia fragmentation pre-processing phase introduced in [Saad *et al.* 2006], which is to be executed prior to applying the classic fragmentation algorithms [Ozsu and Valduriez 1991], [Navathe *et al.* 1995]

5.1. Value Implication

A value V_i implies V_j if the corresponding value concepts Vc_i and Vc_j are such as the global neighborhood of Vc_i includes that of Vc_j in the used value knowledge base:

$$V_i \Rightarrow V_j \quad If \quad \overline{N_{V_{KB}}}(Vc_j) \subset \overline{N_{V_{KB}}}(Vc_i) \qquad \textbf{(3)}$$

Note that when V_i and V_j are synonyms, that is when Vc_i and Vc_j designate the same *value concept* (e.g. *car* and *automobile*), implication exists in both directions: $V_i \Rightarrow V_j$ and $V_j \Rightarrow V_i$. Known as *equivalence implication*, it is designated as $V_i \Leftrightarrow V_j$.

$$V_i \Leftrightarrow V_j \qquad If \qquad \overline{N_{V_{KB}}}(Vc_i) = \overline{N_{V_{KB}}}(Vc_j)\text{ , i.e. } Vc_i \text{ and } Vc_j \text{ are the same} \qquad \textbf{(4)}$$

Our *Value_Implication* algorithm is developed in Figure 4. The algorithm returns values comprised in {0, -1, 1, 2} where:

- '0' denotes the implication absence between the compared values,
- '-1' designates that value V_j implies V_i,
- '1' designates that value V_i implies V_j,
- '2' designates that values V_i and V_j are *equivalent.*

A special case of value implication to be considered is when sets of values are utilized in multimedia predicates. This occurs when set operators come to play (e.g. *Keywords = ANY {"Eiffel Tower", "Coliseum"}* and *Keywords = ANY {"Paris", "Rome"}*). The algorithm for determining the implication between two sets of values is developed in Figure 6. It considers each set of values in isolation and, for each value in the set, computes the neighborhood of the value. Subsequently, it identifies the union of all the neighborhoods of values for the current set (cf. Figure 6, lines 1-7), and compares the 'unioned' neighborhoods of the two sets being treated so as to determine the implication (cf. Figure 6, lines 8-17). In other words, when comparing sets VS_1 and VS_2:

- If $|VS_1| < |VS_2|$ and all values of VS_2 imply (or are equivalent to) those of VS_1, then the set VS_2 implies VS_1 (i.e. the neighborhood of VS_2 includes that of VS_1).
- If $|VS_1| > |VS_2|$ and all values of VS_1 imply (or are equivalent to) those of VS_2, then the set VS_1 implies VS_2 (i.e. the neighborhood of VS_1 includes that of VS_2).
- Otherwise if $|VS_1| = |VS_2|$, then:
  - VS_1 is equivalent to VS_2 when all values of VS_1 are equivalent to those of VS_2 (i.e. the neighborhoods of VS_1 and VS_2 are identical).
  - VS_1 implies VS_2 when all values of VS_1 imply those of VS_2, i.e. the neighborhood of VS_1 encompasses that of VS_2: $\overline{N_{V_{KB}}}(VS_2) \subset \overline{N_{V_{KB}}}(VS_1)$
  - VS_2 implies VS_1 when all values of VS_2 imply those of VS_1, i.e. $\overline{N_{V_{KB}}}(VS_1) \subset \overline{N_{V_{KB}}}(VS_2)$
  - Otherwise, there is no implication between VS_1 and VS_2.

For example, applying *Value Set implication* to sets VS_1 = *{"Eiffel Tower", "Coliseum"}* and VS_2 = *{"Paris", "Rome"}* yields $VS_1 \Rightarrow VS_2$ having:

- $|VS_1| = |VS_2|$
- all values of VS_1 imply those of VS_2: *Eiffel Tower* $\Rightarrow$ *Paris* and *Coliseum* $\Rightarrow$ *Rome* (cf. Figure 1).

5.2. Operator Implication

Similarly to values, the general implication concept remains unchanged with operators. An operator θ_i implies θ_j ($\theta_i \Rightarrow \theta_j$) if the corresponding operator concepts Oc_i and Oc_j are such as the global neighborhood of θ_i includes that of θ_j, following the operator knowledge base defined in Section 4.1.2. We formally write it as:

$$\theta_i \Rightarrow \theta_j \quad If \quad \overline{N_{O_{KB}}}(Oc_j) \subset \overline{N_{\theta_{KB}}}(Oc_i) \tag{5}$$

Similarly to value implication, when θ_i and θ_j are synonyms (e.g. *=any* and *=some* following θ_{KB}), *equivalence* implication exists in both directions:

$$\theta_i \Leftrightarrow \theta_j \quad If \quad \overline{N_{O_{KB}}}(Oc_i) = \overline{N_{O_{KB}}}(Oc_j)\text{, i.e. } Oc_i \text{ and } Oc_j \text{ are the same} \tag{6}$$

The *Operator_Implication* algorithm is developed in Figure 5. It returns values comprised in {0, -1, 1, 2}:

- '0' denoting the lack of implication between the operators' values,
- '-1' designating that operator θ_j implies θ_i,
- '1' designating that operator θ_i implies θ_j,
- '2' when operators θ_i and θ_j are equivalent

5.3. Predicate Implication

$$\mathrm{P}_i \Rightarrow \mathrm{P}_j \;\; if \begin{bmatrix} \theta_i \Rightarrow \theta_j \;\text{ and }\; V_i \Rightarrow V_j \;, or \\ \theta_i \Leftrightarrow \theta_j \;\text{ and }\; V_i \Rightarrow V_j \;, or \\ \theta_i \Rightarrow \theta_j \;\text{ and }\; V_i \Leftrightarrow V_j \end{bmatrix} \tag{7}$$

Let $P_i = A_i\ \theta_i\ V_i$ and $P_j = A_j\ \theta_j\ V_j$ be two predicates employing comparison or set operators. The implication between P_i and P_j, denoted as $P_i \Rightarrow P_i$, occurs if the operator and value (set of values) of P_i (θ_i and V_i) respectively imply those of P_j (θ_j and V_j), or the value (set of values) part of P_i (V_i) implies that of P_j (V_j) when having equivalent operators.

When both pairs of values (sets of values) and operators are equivalent, the corresponding predicates are equivalent as well:

$$\mathrm{P}_i \Leftrightarrow \mathrm{P}_j \;\; if \left[\theta_i \Leftrightarrow \theta_j \text{ and } V_i \Leftrightarrow V_j\right] \quad \textbf{(8)}$$

Value Implication:

Input: V_i , V_j , V_{KB} // V_{KB} is the reference value KB.
Output: {0, -1, 1, 2} // A numerical value indicating
// if $V_i \not\Leftrightarrow V_j$ (0), $V_j \Rightarrow V_i$ (-1) ,
// if $V_i \Rightarrow V_j$ (1) or if $V_i \Leftrightarrow V_j$ (2)

Begin
  If ($\overline{N_{V_{KB}}}(Vc_i) = \overline{N_{V_{KB}}}(Vc_j)$)
    Return 2 // synonyms, $V_i \Leftrightarrow V_j$
  Else If $\overline{N_{V_{KB}}}(Vc_j) \subset \overline{N_{V_{KB}}}(Vc_i)$
    Return 1 // $V_i \Rightarrow V_j$
  Else If $\overline{N_{V_{KB}}}(Vc_i) \subset \overline{N_{V_{KB}}}(Vc_j)$
    Return -1 // $V_j \Rightarrow V_i$
  Else
    Return 0 // There is no implication
  End If // between V_i and V_j, $V_i \not\Leftrightarrow V_j$
End

Figure 4. Identifying semantic implications between textual values

Operator Implication:

Input: θ_i , θ_j , O_{KB} // O_{KB} is the reference operator KB
Output: {0, -1, 1, 2} // A numerical value indicating
// if $\theta_i \not\Leftrightarrow \theta_j$ (0), if $\theta_j \Rightarrow \theta_i$ (-1)
// if $\theta_i \Rightarrow \theta_j$ (1), or if $\theta_i \Leftrightarrow \theta_j$ (2)

Begin
  If($\overline{N_{O_{KB}}}(Oc_i) = \overline{N_{O_{KB}}}(Oc_j)$)
    Return 2 // synonyms, $\theta_i \Leftrightarrow \theta_j$
  Else If $\overline{N_{O_{KB}}}(Oc_j) \subset \overline{N_{O_{KB}}}(Oc_i)$
    Return 1 // $\theta_i \Rightarrow \theta_j$
  Else If $\overline{N_{O_{KB}}}(Oc_i) \subset \overline{N_{O_{KB}}}(Oc_j)$
    Return -1 // $\theta_j \Rightarrow \theta_i$
  Else
    Return 0 // There is no implication between
  End If // θ_i and θ_j, $\theta_i \not\Leftrightarrow \theta_j$
End

Figure 5. Identifying implications between operators

Value Set Implication:

Input: VS_1, VS_2, V_{KB} // value sets to be compared w.r.t. V_{KB}
Output: {0, -1, 1, 2} // A numerical value indicating
// if $VS_1 \not\Leftrightarrow VS_2$ (0), if $VS_2 \Rightarrow VS_1$ (-1)
// if $VS_1 \Rightarrow VS_2$ (1) or if $VS_1 \Leftrightarrow VS_2$ (2)

Begin
  For each value V_i in VS_1 // Neighborhood of VS_1
    $\overline{N_{V_{KB}}}(VS_1) = \overline{N_{V_{KB}}}(VS_1) \cup \overline{N_{V_{KB}}}(Vc_i)$
  End for
  For each value V_j in VS_2 // Neighborhood of VS2
    $\overline{N_{V_{KB}}}(VS_2) = \overline{N_{V_{KB}}}(VS_2) \cup \overline{N_{V_{KB}}}(Vc_j)$
  End For
  If $\overline{N_{V_{KB}}}(VS_1) = \overline{N_{V_{KB}}}(VS_2)$
    Return 2 // $VS_1 \Leftrightarrow VS_2$
  Else If $\overline{N_{V_{KB}}}(VS_2) \subset \overline{N_{V_{KB}}}(VS_1)$
    Return 1 // $VS_1 \Rightarrow VS_2$
  Else If $\overline{N_{V_{KB}}}(VS_1) \subset \overline{N_{V_{KB}}}(VS_2)$
    Return -1 // $VS_2 \Rightarrow VS_1$
  Else
    Return 0 // There's no implication
  End If // between VS_1 and VS_2, $VS_1 \not\Leftrightarrow VS_2$
End

Figure.6 Value sets implication algorithm

Our *Semantic Predicate Implication* (*SPI*) algorithm, developed in Figure 7, utilizes the preceding rules to generate the semantic predicate Implications Set (*IS)* for a given multimedia type. The implications are designated as doublets *($P_i \Rightarrow P_j$)*. Note that in *SPI*, the input parameters of *Value_Implication* and *Value_Set_Implication* between brackets, i.e. V_i and V_{i+1}, designate single values and set values respectively following the considered predicate (cf. Definition 4).

```
Semantic Predicate Implication (SPI):

Input: P , V_KB, O_KB          // P is the set of predicates utilizing semantic operators,
                               // applied on a given multimedia type to be fragmented.
Output: IS                     // Set of semantic predicate implications.
Variables: Implication_Operator , Implication_Value

Begin
  For each P_i in P
      For each P_i+1 in P

              Implication_Operator = Operator_Implication(θ_i , θ_i+1, O_KB)
              If (θ_i , θ_i+1 ε { θ_c any, θ_c some, θ_c all, In}          // Set operators
                  Implication_Value = Value_Set_Implication (V_i , V_i+1, V_KB)
              Else                                                        // Mono-valued operators
                  Implication_Value = Value_Implication(V_i , V_i+1, V_KB)
              End If

             If (Implication_Operator == 2)                  // θ_i ⇔ θ_i+1

                        If (Implication_Value == 2)           // V_i ⇔ V_j
                              IS = IS ∪ (P_i ⇔ P_j)
                        Else If (Implication_Value == 1)      // V_i ⇒ V_j
                              IS = IS ∪ (P_i ⇒ P_j)
                        Else If (Implication_Value == -1)     // V_j ⇒ V_i
                              IS = IS ∪ (P_j ⇒ P_i)
                        End If

                Else If (Implication_Operator == 1)          // θ_i ⇒ θ_j

                        If(Implication_Value == 2 or Implication_Value == 1)      // θ_i ⇒ θ_j
                              IS = IS ∪ (P_i ⇒ P_j)
                        End If

                Else If (Implication_Operator == -1)         // θ_j ⇒ θ_i

                        If (Implication_Value == 2 or Implication_Value == -1)    // V_j ⇒ V_i
                              IS = IS ∪ (P_j ⇒ P_i)
                        EndIf
                End If
      End For
  End For
End
```

Figure 7. Algorithm *SPI* for identifying the semantic implications between predicates

5.4. Algorithm Complexity

The computational complexity of our *Semantic Predicate Implication (SPI)* is estimated on the basis of the worst case scenario. Suppose n_c represents the number of concepts in the concept knowledge base considered, d the maximum depth in the concept knowledge base considered, n_{pv} the number of user predicates with single values, n_{pvs} the number of predicates with value sets, and n_v the maximum number of values contained in a value set. *SPI* algorithm is of time complexity $O(n_{pv}^2 \times n_c \times d + n_{pvs}^2 \times n_v \times n_c \times d)$ since:

- The neighborhood of a concept is generated in $O(n_c \times d)$ time, which comes down to the complexity of algorithm *Value_Implication*.
- The neighborhood of an operator is generated in constant time: $O(1)$, which comes down to the time complexity of algorithm *Operator_Implication*. Therefore, identifying implications for predicates with simple values is of time complexity $O(n_{pv}^2 \times n_c \times d)$.
- The *Value_Set_Implication* algorithm is of complexity $O(n_v \times n_c \times d)$

Subsequently, identifying semantic implications for predicates with value sets is of time complexity $O(n_{pvs}^2 \times n_v \times n_c \times d)$.

6. Implementation and Experimental Tests

6.1. Prototype

To validate our approach, we have implemented a C# prototype entitled "Multimedia Semantic Implication Identifier" (MSI^2) encompassing:

- A relational database, storing multimedia objects via Oracle 9i DBMS,
- Relational tables for storing the reference value knowledge base V_{KB} and the operator knowledge base O_{KB}. Note that O_{KB} is constant (cf. Figure 2),
- An interface allowing users to formulate multimedia queries.

In Figure 8, we show how the prototype accepts a set of input multimedia queries. Automatic processes subsequently calculate query access frequencies, identify corresponding predicates, and compute for each multimedia type (cf. Definition 2) its Predicate Usage Matrix (PUM) and its Predicate Affinity Matrix (PAM), introduced in [Navathe *et al.* 1995], [Belatreche *et al.* 1997] (cf. Figure 8). The PAM is used to underline the affinity between predicates, *implication* being a special kind of affinity [Navathe *et al.* 1995], [Belatreche *et al.* 1997]. The PUM and PAM make up the inputs to the primary horizontal partitioning algorithm: *Make_Partition* [Navathe *et al.* 1995] or *Com_Min* [Ozsu and Valduriez 1991].

6.2 Simulation Example

Among the various experiments conducted, we present here a simple simulation example comparing predicate affinities (PAM) obtained with the inclusion of our multimedia semantic implication rules, and analyzing the corresponding fragments. In the following example, multimedia type "Video", designating movies (i.e. audio-visual data), is selected for PUM and PAM calculations. In this experiment, a 100 node knowledge base, extracted from WordNet provided the reference for *predicate value* implications (part of the knowledge base is depicted in Figure 1). Let Q = $\{Q_{i = 0 \text{ to } 5}\}$ be a set of user queries searching for video objects and P = $\{P_{i = 0 \text{ to } 11}\}$ be the set of predicates used by Q (Figure 8). Given the PUM, the PAM attained after applying our semantic implication algorithms in shown in Figure 8. Note that the traditional PAM matrix will lack our semantic implications, identified here by implication signs, and only contains null affinities instead (it is omitted due to the lack of space).

Recall that following [Navathe *et al.* 1995] [Belatreche *et al.* 1997], 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[1] underlines the implication relation between predicates P_i and P_j. Note that "numerical" predicates, yielding traditional implications (for example P_1: $x < 2 \Rightarrow P_2$: $x < 4$), were excluded for the sake of simplicity and clearness. Hence, the traditional PAM should be restricted to numerical affinities whereas the updated PAM should reflect both numerical and non numerical (semantic implication) affinities:

- Predicates P_5 (Event = "Football match") and P_7 (Event = "Hokey match") imply P_0 (Event = "Sport game").
- P_1 (Location = "Rome"), P_{10} (Location = "Paris") and P_{12} (Location = "Champs Elysees") imply P_6 (Location *like* "%Europe") having:
  - = $\Rightarrow$ *like %* (cf. Figure 2).
  - Rome, Paris, Champs Elysees $\Rightarrow$ Europe.
- Predicate P_{12} (Location = "Champs Elysees") implies P_{10} (Location = "Paris").

[1] Non numerical affinity can also designate the "close" usage of two predicates P_i and P_j, in that both P_i and P_j are used jointly with a predicate P_l [15]. Nonetheless, we disregard this kind of affinity for the sake of clearness.

- Predicate P_4 (Keywords ≠ *all* ("Night", "Freeway", "Speed")) implies P_2 (Keywords ≠ "Night"):
  - ≠ *all* ⇒ ≠ (cf. Figure 2).
  - (Night, Freeway, Speed) ⇒ Night.
- Predicate P_9 (Event = "Car crash") implies P_3 (Event like "_Accident") having:
  - = ⇒ *like* _ (cf. Figure 2).
  - Car crash ⇒ Accident
- Predicate P_{11} (Keywords = *Some* ("Night", "Freeway", "Speed")) implies P_{13} (Keywords *in* ("Highway", "Night")) having:
  - = *Some* ⇒ *in* (cf. Figure 2).
  - (Night, Freeway, Speed) ⇒ (Highway, Night)

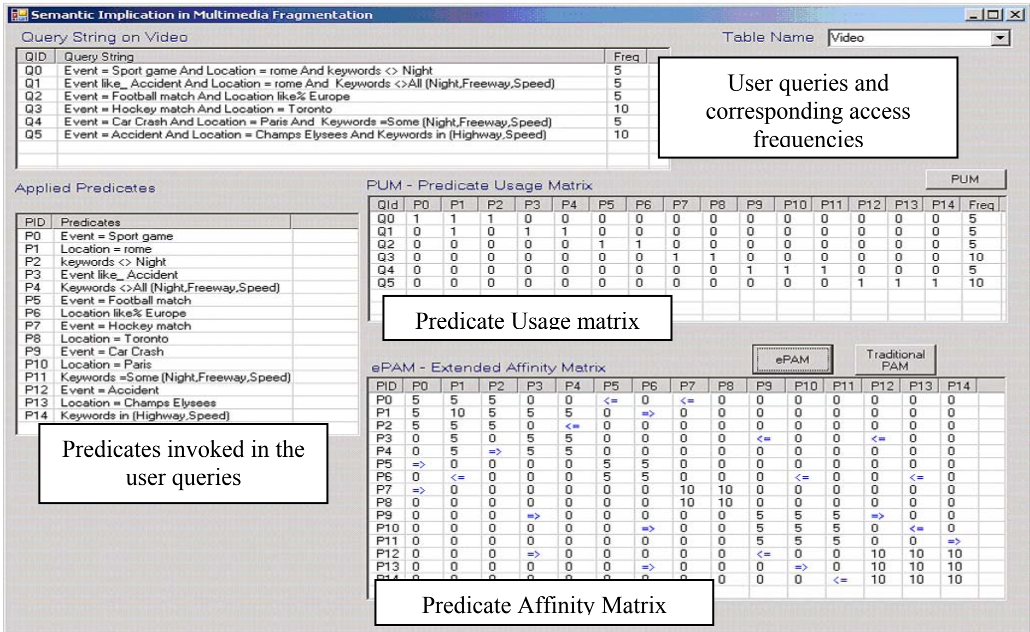


Figure 8. Screen shot of the *MSI*^{2} PUM and PAM interface.

The primary horizontal fragmentation algorithm *Make-Partition* [Navathe *et al.* 1995], applied on the uPAM matrix obtained above, generates the predicate clusters shown in Figure 9. These clusters are further refined following a post-processing procedure developed in [Belatreche *et al.* 1997], based on the semantic implications identified in the uPAM, to yield the final horizontal minterm fragments shown in Figure 9. As a matter of fact, since $P_4 \Rightarrow P_2$, $P_9 \Rightarrow P_3$, $P_{13} \Rightarrow P_{10}$ and $P_{11} \Rightarrow P_{14}$, then P_4, P_9, P_{13} and P_{11} should be removed from the corresponding clusters [Belatreche *et al.* 1997], consequently yielding the minterms shown below.

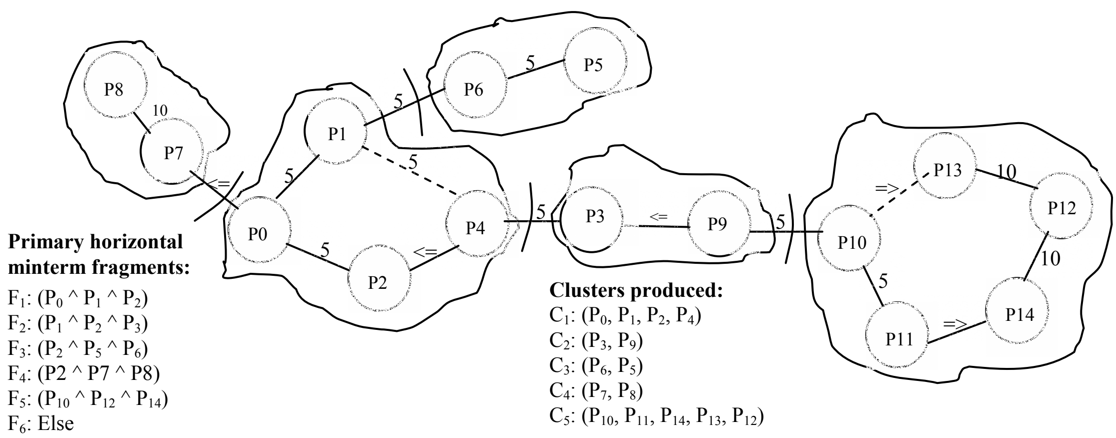
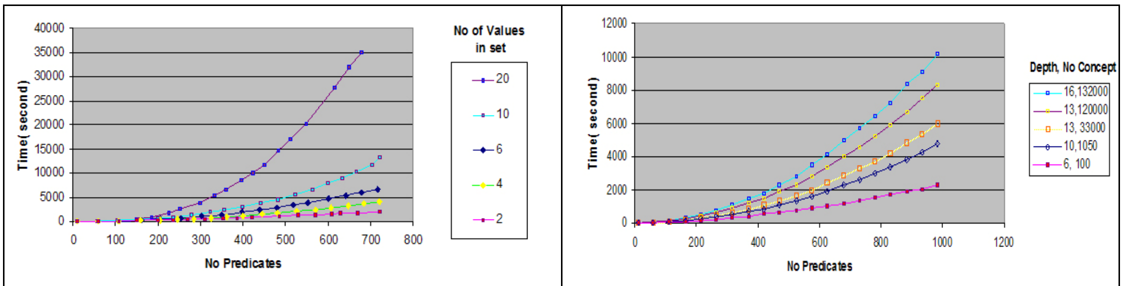


Figure 9. Clustering of Predicates using graph theoretic algorithm

Recall that ignoring implications can lead, in multimedia applications, to higher computation costs when creating fragments, bigger fragments which are very restrictive for multimedia storage, and retrieval, as well as data duplication on several sites. For instance, in the current example, applying *Make_Partition* without considering the semantic implications between predicates (PAM lacking all semantic implications, which are replaced by null values) yields the following minterm fragments: $F_1(P0 \wedge P1 \wedge P2)$, $F_2(P1 \wedge P2 \wedge P3)$, $F_3(P0 \wedge P1 \wedge P4)$, $F_4(P2 \wedge P5 \wedge P6)$, $F_5(P2 \wedge P7 \wedge P8)$, $F_6(P1 \wedge P2 \wedge P9)$, $F_7(P0 \wedge P10 \wedge P11)$, $F_8(P12 \wedge P13 \wedge P14)$, *F9(Else)*. On can clearly recognize the higher number of minterms, in comparison with those identified using the semantic implications (i.e., uPAM in Figure 8), which obviously underlines higher computation costs when creating the multimedia partitions. In addition, the obtained fragments induce data duplication among each other, e.g., between F_1 and F_3, as well as F_2 and F_6, which is detrimental to data fragmentation.

6.2. Timing Analysis

We have shown that the complexity of our approach (*SPI* and underlying algorithms) simplifies to $O(n_{pvs}^2 \times n_v \times n_c \times d)$. It is quadratic in the size of user predicates (n_{pvs}^2), and varies with value set cardinalities (n_v), as well as the size of the value knowledge base V_{KB} considered ($n_c \times d$). We have verified those results experimentally. Timing analysis is presented in Figure 10. The experiments were carried out on Pentium 4 PC (with processing speed of 3.0 GHz, 504 MB of RAM). Note that in these experiments, a set of 1200 semantic predicates was generated in a random fashion, value-set cardinalities (varying between 2 and 20 per value set, cf. Figure 10) being under strict user control. Multiple value knowledge bases, extracted from WordNet, with varying depth (from 6 to 16 levels, cf. Figure 10.b) and number of concepts (from 100 to 132000 nodes, cf. Figure 9.b) were also considered. One can see from the result that the time to compute semantic implications grows in a polynomial fashion with the number of predicates.



a. Varying value set cardinalities   **b.** Varying V_{KB} size (depth and number of concepts)

Figure 10. Timing results regarding the number of predicates, value set cardinalities, and V_{KB} size

Recall that the reference *value knowledge base* V_{KB} and *operator knowledge base* O_{KB} are stored in a relational database and are queried for each value and operator in the concerned predicates when identifying implication. Thus, querying the V_{KB} knowledge base for each predicate value requires extra time (database access time) and hence contributes to increasing time complexity. Therefore, we believe that system performance would improve if the reference V_{KB} knowledge base could fit in primary memory.

7. Conclusion

Fragmentation techniques are used in distributed system design to reduce accesses to irrelevant data, thus enhancing system performance [Ezeife and Barker 1995]. In this study, we address primary horizontal fragmentation in multimedia databases. In particular, we emphasize semantic-based predicates implication which are required in current fragmentation algorithms, in order to partition multimedia data efficiently. Our approach is complementary to that developed in [Saad *et al.* 2006], targeting implications between low-level multimedia predicates (applied on complex feature vectors such as dominant color, texture, etc.) as a prerequisite to performing multimedia fragmentation. We propose a set of algorithms for identifying implications between semantic predicates, based on operator and value implications. Operator implications are identified utilizing a specific operator knowledge base developed in our study. Value implications are discovered following domain-oriented or generic value concept knowledge bases such as WordNet [WordNet 2005]. We developed a prototype to test our approach. Timing results show that our method is of polynomial complexity.

We are currently conducting fragmentation experiments on real multimedia data so as to analyze our approach's efficiency with respect to traditional methods. Future directions include studying derived horizontal fragmentation and vertical fragmentation of multimedia data, taking into account semantic and low-level multimedia features. We also plan on releasing a public version of our prototype.

References

Baiao F, Mattoso M., A Mixed Fragmentation Algorithm for Distributed Object Oriented Databases. *9th Inter. Conf. on Computing Information*, Canada, 1998

Belatreche L, Karlapalem K, Simonet A., Horizontal class partitioning in object-oriented databases. *8th Inter. Conf. on Database and Expert Systems Applications* (*DEXA'97*), 1997

Bernhard Braunmuller, Efficiently Supporting Multiple Similarity Queries for Mining in Metric Databases, *IEEE Trans. on Knowledge and Data Engineering,* v.13, p.79-95, 2001

Chinchwadkar G.S., Goh A., An Overview of Vertical Partitioning in Object Oriented Databases. *The Computer Journal,* Vol. 42, No. 1, 1999

Ehrig M. and Sure Y., Ontology Mapping - an Integrated Approach. In *Proceedings of the 1st European Semantic Web Symposium*, V. 3053 of LNCS, pp. 76-91, Greece, 2004

Ezeife C.I., Barker K., A Comprehensive Approach to Horizontal Class Fragmentation in a Distributed Object Based System. *J. of Distributed and Parallel Databases*, 1, 1995.

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

Gertz M, Bremer J.M., Distributed XML Repositories: To-Down Design and Transparent Query Processing. Department of CS, University of California, 2004

Grosky W. I., Managing Multimedia Information in Database Systems, *Communications of the ACM*, Vol. 40, No. 12, pp. 72-80, 1997

Kosch H., Distributed Multimedia Database Technologies Supported by MPEG-7 and MPEG-21, Auerbach Publications, 280 p., 2004

Lin D., An Information-Theoretic Definition of Similarity. *In Proceedings of the 15th International Conference on Machine Learning*, 296-304, 1998.

Maguitman A. G., Menczer F., Roinestad H. and Vespignani A., Algorithmic Detection of Semantic Similarity. In *Proc. of the 14th Inter. WWW Conference*, 107-116, Japan, 2005

Miller G., WordNet: An On-Line Lexical Database. *Journal of Lexicography*, 3(4), 1990.

Navathe B, RA M., Vertical Partitioning for Database Design: a Graphical Algorithm. *1989 ACM SIGMOND Conference*, Portland, 440-450, 1989

Navathe S.B, Karlapalem K, Ra M., A Mixed Partitioning Methodology for Initial Distributed Database Design. *Computer and Software Engineering J.*, 3(4): 395-426, 1995

Ozsu M.T, Valduriez P., *Principals of Distributed Database Systems*, Prentice Hall, 1991

Richardson R. and Smeaton A.F., Using WordNet in a Knowledge-based approach to information retrieval. In *Proc. of the 17th Colloquium on Information Retrieval*, 1995.

Rodriguez M.A., Egenhofer M.J., Determining Semantic Similarity among Entity Classes from Different Ontologies. *IEEE Transactions on Knowledge and Data Engineering*, Vol.15, n.2, pp. 442-456, 2003

Saad S., Tekli J., Chbeir R. and Yetongnon K., Towards Multimedia Fragmentation. In *Proceedings of the 10th East-European Conference on Advanced Databases and Information Systems ADBIS'06*, 2006

Sub C., An approach to the model-based fragmentation and relational storage of XML-documents. Grundlagen von Datenbanken, 98-102, 2001

WordNet 2.1, A Lexical Database of the English Language. http://wordnet.princeton.edu/online/, 2005.