The Use of Semantic-based Predicates Implication to Improve Horizontal Multimedia Database Fragmentation

Fekade Getahun, Solomon Atnafu
Department of Computer Science, Faculty of Informatics
Addis Ababa University, 1176 Addis Ababa, Ethiopia
{fekadeg, satnafu}@cs.aau.edu.et

Joe Tekli, Richard Chbeir
LE2I-CNRS Laboratory, University of Bourgogne
21078 Dijon Cedex France
{joe.tekli, richard.chbeir}@u-bourgogne.fr

ABSTRACT
Database fragmentation allows reducing irrelevant data accesses by grouping data frequently accessed together in dedicated segments. In this paper, we address multimedia database fragmentation to take into account the rich characteristics of multimedia objects. We particularly discuss multimedia primary horizontal fragmentation and focus on semantic-based textual predicates implication required as a pre-process in current fragmentation algorithms in order to partition multimedia data efficiently. Identifying semantic implication between similar queries (if a user searches for the images containing a car, he would probably mean auto, vehicle, van or sport-car as well) will improve the fragmentation process. Making use of the neighborhood concept in knowledge bases to identify semantic implications constitutes the core of our proposal. A prototype has been implemented to evaluate the performance of our approach.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Storage – *Record Classification;* Information Search and Retrieval – *Search Process*; H.2.7 [**Database Management**]: Database Administration*;* H.2.8 [**Database Management**]: Database Applications; H.2.5 [**Database Management**]: Heterogeneous Databases; H.2.4 [**Database Management**]: Systems.

General Terms
Algorithms, Measurement, Performance, Design, Experimentation.

Keywords
Multimedia Retrieval, Horizontal Fragmentation, Data Partition, Data implication

1. INTRODUCTION
Multimedia applications emerging in distributed environments, such as the web, 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 DataBase Management Systems MMDBMS [8]. Nevertheless, most existing systems lack a formal framework to adequately provide full-fledge multimedia operations. Traditionally, fragmentation techniques are used in distributed system design to reduce accesses to irrelevant data, thus enhancing system performance [4]. 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. While partitioning traditional databases has been thoroughly studied, multimedia fragmentation has not yet received strong attention. 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* [1, 13, 14], in order to partition multimedia data efficiently. The need of such semantic-based implication is emphasized by the fact that annotations and values describing the same object, during the storage or retrieval of multimedia data, could be interpreted with largely different meanings. For example, if a user searches for the images containing a *car*, he would probably mean *auto, vehicle, van or sport-car* as well. Therefore, it is obvious that semantic implication between such similar values will improve the fragmentation process (and more particularly will impact the choice of *minterms* as we will see in the remaining sessions). The contribution of the paper can be summarized as follows: i) introducing algorithms for identifying semantic implications between predicate values, ii) introducing an algorithm for identifying semantic implications based on predicate operators, iii) putting forward an algorithm for identifying implications between semantic predicates on the basis of operator and value implications, iv) developing a prototype to test and validate our approach.

The remainder of this paper is organized as follows. Section 2 briefly reviews the background and related work in DB fragmentation. In Section 3, we present a motivation example. Section 4 is devoted to define the concepts to be used in our approach. In Section 5, we detail our semantic implication algorithms and their usage in the multimedia fragmentation process. Section 6 briefly presents our prototype. Finally, Section 7 concludes this work and draws some ongoing research directions.

2. BACKGROUND AND RELATED WORK
Fragmentation techniques for distributed DB systems aim to achieve effective resource utilization and improved performance [20]. This is addressed by removing irrelevant data accessed by applications and by reducing data exchange among sites [21]. 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 [14]. 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 [21]. MF is a hybrid partitioning technique where horizontal and vertical fragmentations are simultaneously applied on an entity/class [13].

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 [14] and *Make_Partition* Graphical Algorithm developed by Navathe *et al.* [12] (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 [1] 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 [13] (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 Ezeife and Barker [4] using Com_Min [14], and the other developed by Bellatreche *et al.* [1] on the basis of Make_Partition [12]. The use of Com_Min or Make_Partition is the major difference between them.

Recent works have addressed XML fragmentation [18], [6] 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.

One recent approach is provided by Saad *et al.* in [17] to address multimedia database fragmentation. 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 [17]. 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 [17].

In [1, 13], the authors have highlighted the importance of implication, but have not detailed the issue. As mentioned before, the authors in [17] have only addressed implications between low-level multimedia predicates (based on complex feature vectors). In this study, we go beyond low-level features provided in [17] and present a complementary semantic implication approach

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

Def. 1 - Multimedia Object: is depicted as a set of attribute (a_i) and value (v_i) doublets: O $\{(a_1, v_1), (a_2, v_2), \ldots , (a_n, v_n)\}$. 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).

Def. 2 - Multimedia Type: allocates a set of attributes used to describe multimedia objects corresponding to that type [17]. Two objects, described by the same attributes, are of the same type.

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

Def. 3 - Multimedia Query: is written as follows [1, 17]: 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 $\wedge, \vee, \neg$.

Def. 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 [9, 15], generic concepts [3, 16], web pages [10], etc.) into a semantic space. In our approach, we employ knowledge bases as a reference for identifying semantic implications between predicates. 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 [9, 15, 19], to organize entities and concepts in a KB. We detail them below.

4.2.1 Value 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[2].

As in WordNet[3], 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 $E \subseteq Vc \times Vc$
- 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, $f{:}E \rightarrow R$.

4.2.2 Operator Knowledge Base

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 (=, ≠, >, <[4], 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 $E \subseteq Oc \times Oc$.
- R is the set of semantic relations, $R=\{\Omega, \prec, \succ, \ll, \gg\}$.
- f is a function designating the nature of edges in E, $f{:}E \rightarrow R$ (cf. Figure 1).

---

[2] However, the building process of the value knowledge base is out of the scope of this paper.

[3] WordNet is an online lexical reference system (taxonomy), where nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing a lexical concept [11, 19].

[4] ≥ and ≤ are considered as single operators put together using the Boolean operator OR.

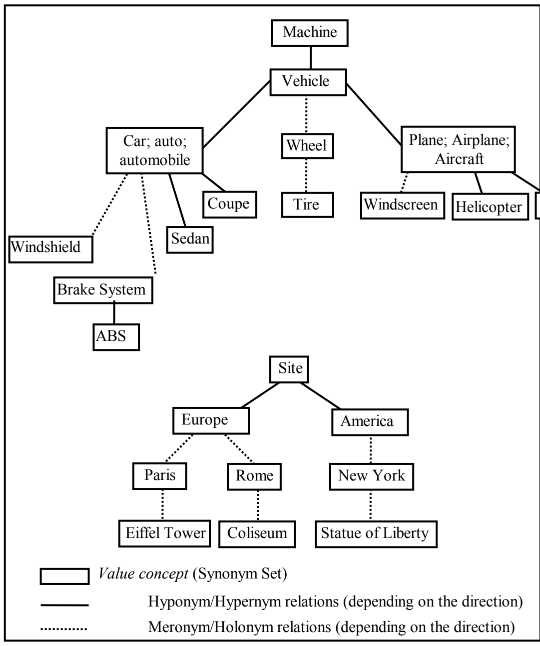


Figure 1. Sample value knowledge base with multiple root concepts

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

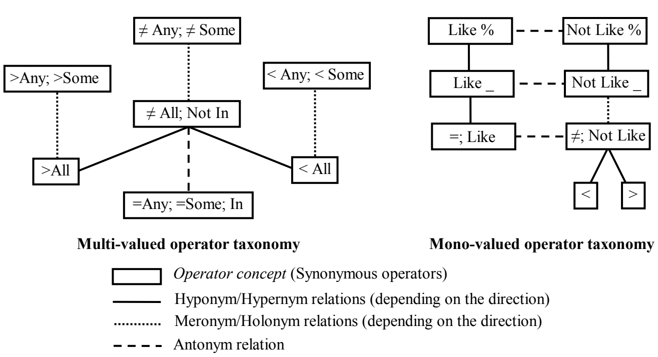


Figure 2. Our proposed operator knowledge base

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[5] following these operators, i.e. *Like_* $\prec$ *Like%* 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.

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:

- *Synonym (≡)*: Two words/expressions/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 [9, 15, 19]. 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.

[5] Relations will be detailed in the next subsection.

4.4 Neighborhood

In our approach, the neighborhood notion is used to compute the implication between values, operators, and consequently predicates.

Table 2. Semantic relations

Relation \ Property	Symbol	Reflexive	Symmetric	Transitive
Synonym	≡	✓	✓	✓
Antonym	Ω	×	✓	×
Hyponym	≺	✓	×	✓
Hypernym	≻	✓	×	✓
Meronym	≪	✓	×	✓
Holonym	≫	✓	×	✓

Table 3. Transitivity between relations

R_{ij} \ R_{jk}	≡	Ω	≺	≻	≪	≫
≡	≡	Ω	≺	≻	≪	≫
Ω	Ω	≡	Ω	Ω	∅	∅
≺	≺	Ω	≺	∅	≪	∅
≻	≻	Ω	∅	≻	∅	≫
≪	≪	∅	≪	∅	≪	∅
≫	≫	∅	∅	≫	∅	≫

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 (≡), hyponym ($\prec$) and meronym ($\ll$) semantic relations, directly or via transitivity. It is formally defined as:

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

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

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

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

Note hereunder the corresponding global neighborhoods of the same examples:

- $\overline{N_{V_{KB}}}(ABS) = \{ABS, brake\ system, car, auto, automobile, vehicle, machine\}$
- $\overline{N_{V_{KB}}}(car) = N^{\equiv}_{V_{KB}}(car) \cup \quad \cup \quad car, auto, automobile, vehicule, machine\}$

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

As finding implication between predicates is crucial to cutback the number of predicates involved in the fragmentation process [1, 15], 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. Therefore, we develop value and operator implications before introducing our predicate implication algorithm. Our *Semantic Implication Algorithm* (*SPI*) is complementary to that developed in [17] 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.

```
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 [17], which is to be executed prior to applying the classic fragmentation algorithms

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 $V_i \Leftrightarrow V_j$.

$$V_i \Leftrightarrow V_j \quad If \quad \overline{N_{V_{KB}}}(Vc_i) = \overline{N_{V_{KB}}}(Vc_j) \qquad \textbf{(4)}$$

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, 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:

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

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

$$\theta_i \Leftrightarrow \theta_j \quad If \quad \overline{N_{O_{KB}}}(Oc_i) = \overline{N_{O_{KB}}}(Oc_j) \qquad \textbf{(6)}$$

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

$$\text{P}_i \Rightarrow \text{P}_j \quad if \quad \begin{bmatrix} \theta_i \Rightarrow \theta_j \ \text{ and } \ V_i \Rightarrow V_j \ , or \\ \theta_i \Leftrightarrow \theta_j \ \text{ and } \ V_i \Rightarrow V_j \ , \ or \\ \theta_i \Rightarrow \theta_j \ \text{ and } \ V_i \Leftrightarrow V_j \end{bmatrix} \qquad \textbf{(7)}$$

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:

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

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).

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

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 $O(n_{pvs}^2 \times n_v \times n_c \times d)$.

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}
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

6. IMPLEMENTATION

6.1 Prototype

To validate our approach, we have implemented a C# prototype entitled "Multimedia Semantic Implication Identifier" (*MSI²*) 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 [1, 15]. The PAM is used to underline the affinity between predicates, *implication* being a special kind of affinity. The PUM and PAM make up the inputs to the primary horizontal partitioning algorithm: *Make_Partition* [15] or *Com_Min* [14].

6.2 Timing Analysis

We have shown that the complexity of our approach (*SPI* and underlying algorithms) simplifies to $O(n_{pvs}^2 \times n_v^2 \times n_c \times d)$. It is quadratic in the size of user predicates (n_{pvs}), value set cardinalities (n_v^2), and 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 9. 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 special process was developed using C# for timing analysis. Large amounts of semantic predicates (that uses our proposed operator knowledge base provided in Section 4.5.2) were generated in a random fashion, predicate numbers as well as value-set cardinalities being under strict user control. Multiple value knowledge bases with varying depth and number of concepts were also considered. Similarity computations and timing analysis were done repeatedly. In both graphs of Figure 9, the x-axis represents the number of predicates and y-axis shows the time needed to compute semantic implications. One can see from the result that the time needed to compute semantic implications grows in a polynomial (quadratic) fashion with the number of predicates involved. Figure 9.a shows the impact of the value-set cardinalities, whereas Figure 9.b reflects the effect of the 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. As a result, 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 overall system performance would improve if the reference V_{KB} knowledge base could fit in primary memory.

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

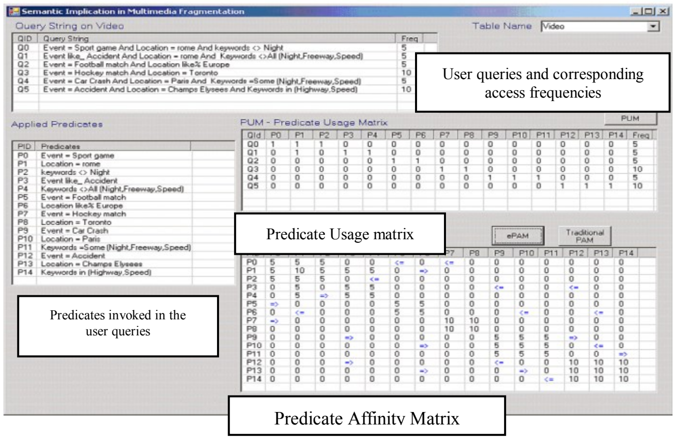
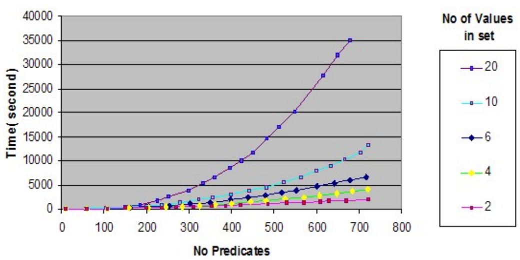
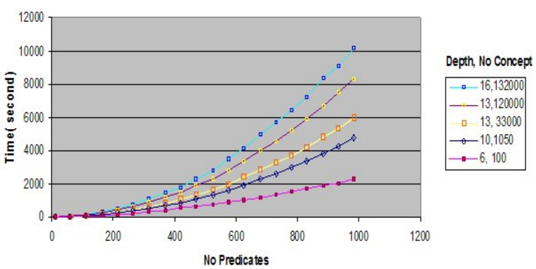


Figure 8. Screen shot of the *MSI²* PUM and PAM interface.



a. Varying value set cardinalities



b. Varying V_{KB} size (depth and number of concepts)

Figure 9. Timing results regarding the number of predicates, value set cardinalities, and V_{KB} size

7. CONCLUSION

In this paper, we addressed the primary horizontal fragmentation in multimedia databases and provided a new approach to be used with existing fragmentation techniques. We particularly studied semantic-based predicates implication required in current fragmentation algorithms in order to partition multimedia data efficiently. We put forward a set of algorithms for identifying implications between semantic predicates on the basis of operator and value implications. Operator implications are identified utilizing a specific operator knowledge base developed in our study. On the other hand, value implications are discovered following domain-oriented or generic value concept knowledge bases such as WordNet. We developed a prototype to test our approach.

In the near future, we aim to thoroughly assess our approach's efficiency via a comparative study so as to show the improvement in fragmentation quality, i.e. the improvement in data access time, w.r.t. existing fragmentation techniques. Our future directions include studying derived horizontal fragmentation of multimedia data, optimizing traditional methods by taking into account semantic and low-level multimedia features. Likewise, multimedia vertical fragmentation and XML fragmentation will also be addressed in upcoming studies. In addition, we plan on releasing a public version of our prototype after integrating low-level multimedia implications.

8. REFERENCES

[1] 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

[2] 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

[3] 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

[4] 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.

[5] 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

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

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

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

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

[10] 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

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

[12] Navathe B, RA M., Vertical Partitioning for Database Design: a Graphical Algorithm. 1989 ACM SIGMOND Conf., Portland, p. 440-450, 1989

[13] 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
[14] Ozsu M.T, Valduriez P., Principals of Distributed Database Systems, Prentice Hall, 1991
[15] 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.
[16] 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
[17] Saad S., Tekli J., Chbeir R. and Yetongnon K., Towards Multimedia Fragmentation. In Proc. of ADBIS'06, Greece, September 2006
[18] Sub C., An approach to the model-based fragmentation and relational storage of XML-documents. Grundlagen von Datenbanken, 98-102, 2001
[19] WordNet 2.1, A Lexical Database of the English Language. http://wordnet.princeton.edu/online/, 2005.
[20] Chinchwadkar G.S., Goh A., An Overview of Vertical Partitioning in Object Oriented Databases. *The Computer Journal,* Vol. 42, No. 1, 1999
[21] Baiao F, Mattoso M., A Mixed Fragmentation Algorithm for Distributed Object Oriented Databases. 9th Inter. Conf. on Computing Information, Canada, 1998