FUZZY QUERY LANGUAGES FOR MULTIMEDIA DATA

Paolo Ciaccia*, Danilo Montesi♦, Wilma Penzo*, and Alberto Trombetta◊

* DEIS-CSITE-CNR, Bologna, Italy, {piaccia, wpenzo}@deis.unibo.it

♦ DSI, *Dept. Of Computer Science*, Milano, Italy, montesi@dsi.unimi.it

◊ DI, *Dept. of Computer Science*, Torino, Italy, tromb@di.unito.it

Abstract

This chapter presents a fuzzy-based algebra, called SAME^W, to query multimedia objects. SAME^W allows for dealing within a common framework with several aspects relevant to *similarity query processing* as well as with the inherent *imprecision* that characterizes data, user requests, and query results. Non-Boolean, namely *fuzzy* and *similarity*, predicates are used to rank tuples according to specific criteria. Complex multi-predicate queries can be formed by means of logical connectives, whose semantics is parameterized in order to adapt to specific scenarios. The same holds for the semantics of algebraic operators. These include properly extended traditional relational operators and new operators which allow *threshold* and *best-matches* queries to be easily expressed. A further important feature of SAME^W is the possibility of *weighting* both predicates and operands of algebraic operators so as to better fit user preferences/requirements. A working example dealing with Web data is used throughout the chapter to show the potentialities of SAME^W. Optimization issues are also briefly discussed.

INTRODUCTION

The advent of the World Wide Web has made available a huge amount of text, image, audio, and video data, collectively referred to as multimedia (MM) data. In recent years, many commercial MM tools have been developed (Flickner, 1995), among which multimedia database systems (MMDBSs) whose aim is to provide unified frameworks for retrieving and integrating MM data (Subrahmanian, 1998). It is a fact that a key feature of MMDBSs should be the capability of expressing highly powerful/complex queries by managing and interpreting the intrinsic imprecision of MM data and by exploiting classification processes that add semantic information to objects.

Many recent works have focused on the design of access methods suitable to index complex features (Ciaccia, 1997; Seidl, 1997), as well as on other performance-related issues, such as approximate queries (Shivakumar, 1998; Ciaccia, 2000-a). On the other hand, more general issues relevant to MM query processing have only been partially addressed. These include models able to capture the "essential" aspects of MM objects needed by *content-based queries*, the impact of *user preferences* on query processing, the management of *imprecise* queries, new kinds of predicates and operators, and so on. Indeed, contributions to above issues (Fagin, 1996; Adali, 1998; Ciaccia, 1998; Montesi, 1999) often consider ad-hoc scenarios and/or completely ignore the other coordinates of the problem, thus resulting in a set of difficult to integrate recipes.

In this chapter we present an extended relational framework that takes into account the two major sources of imprecision arising from queries on MM databases (Soffer, 1998): 1) imprecision of *classification* of MM data, and 2) imprecision in the *matching of features* that characterize the content of MM objects. As to the first point we rely on basic concepts from *fuzzy set theory*, and allow representation of vague classification both at tuple and at attribute level. We then introduce a *similarity algebra*, called SAME^{w} ("Similarity Algebra for Multimedia Extended with Weights"), that extends relational algebra in a conservative way and incorporates the use of *weights* in predicates and operators, in order to model user preferences. We show how complex queries can be easily expressed in SAME^{w} and sketch how equivalence rules can be exploited for the purpose of query rewriting and optimization.

RELATED WORK

Broadly speaking, there are two main approaches to deal with data imprecision. They are based on probability and fuzzy set theories, respectively. Probabilities are simple, yet intuitive to understand. However, the probability of complex (joint) events can be easily computed only under the hypothesis of events' independency. If this hypothesis does not hold (as it often happens), then an interval of probabilities needs to be considered for complex events, as the ProbView system does (Lakshmanan, 1997). Fuzzy concepts do not require such hypothesis since they do not rely on the notion of event. Due to their generality, they have been successfully applied to integrate information from different MM systems (Fagin, 1996).

As for query languages for MM data, specific functionalities have been investigated in recent years. For instance, the modeling of imprecisely classified data was thoroughly analyzed in (Raju, 1988). An interesting approach for "ranking" query results coming from different data sources is proposed in (Gravano, 1997), whereas (Carey, 1997) introduces an operator for selecting only the *n* "best" results of an SQL query. An algebraic setting more suitable for the integration of similarity measures coming from different sources rather than for similarity searches is proposed in (Adali, 1998).

In summary, there are no proposals considering a full-fledged query language expressing the basic MM functionalities within a uniform framework. This language must blend imprecision on data and their classification, specific operators needed to express complex queries on multimedia data, and user preferences. This is the subject developed in the remaining of this chapter.

A WORKING EXAMPLE

As a working example, we consider the repositories of text and images collected in the Web sites of the *Uffizi Gallery* (`www.uffizi.firenze.it`) and of the *Louvre Museum* (`www.louvre.fr`). The user visiting, say, the Uffizi site has several ways to browse through the paintings, accessing them by author's name, time of composition, and geographical provenance, or simply by clicking

on the rooms' map, and then entering the corresponding room where links to the available paintings are collected. However, no search capabilities over the paintings are provided. To this end, let us assume that the stored information about a painting includes ordinary data (such as the title and the author's name), as well as other attributes that describe the *content* of the painting. These can be either *feature attributes* or *fuzzy attributes*.

A feature attribute contains complex data extracted from the painting, such as its color histogram, texture vector, and so on. In order to compare feature values, *similarity predicates* are used to support *content-based retrieval*, e.g.:

(Q1) *"Find paintings with a texture similar to a given input texture vector"*.

When applied to a painting, a similarity predicate returns a *score*, normalized in the interval [0,1], assessing the degree of similarity between the feature value of the painting and the input query value.

A fuzzy attribute stores information obtained from a (possibly manual) *classification process*. Because of the inherent imprecision arising when classifying MM data, a numerical score is also present. This is a membership degree to the *fuzzy set* represented by the value of the fuzzy attribute. We remind that a fuzzy set F over a "universe" U is a set characterized by a membership function $\mu_F : U \rightarrow [0,1]$ where $\mu_F(x)$ is the *degree of membership* of x in F (Klir, 1995). For instance, a painting could be classified as "well preserved" with score 0.8. Note that classification is also possible with respect to properties that can be represented as feature attributes (e.g., a painting could be classified as "red" with score 0.65). A *fuzzy predicate* such as:

(Q2) *"Find paintings in a very good state of conservation"*

returns very well preserved paintings with their classification scores.

Complex queries like:

(Q3) *"Find red paintings with a texture similar to a given input texture vector"*

are evaluated by properly combining the scores corresponding to the single predicates (see next section for details).

When dealing with scores, it is useful to allow the user to assign a distinct *relevance* to the predicates in a complex query. *Weights* are introduced with this aim and provide additional flexibility to express user requirements. They can be attached to the predicates of a complex query so that the resulting score depends more on the scores of the highest weighted predicates. For instance:

(Q4) *"As (Q3). However, color is twice as relevant as texture"*.

Weights are also useful when combining results from different sources, possibly because one trusts more one source with respect to others, e.g.:

(Q5) *"Find paintings in the Uffizi Gallery and in the Louvre Museum that are in a very good state of conservation. Results from the Uffizi Gallery have weight 0.6, whereas those from the Louvre Museum have weight 0.4"*.

In order to limit the cardinality of query results, two basic mechanisms are available. The first one discards all the objects for which the computed score is below a user-specified threshold value, e.g.:

(Q6) *"Find paintings whose texture is similar to a given input texture vector with score at least 0.8"*.

Since setting adequate threshold values can be rather tricky (low values risk to still return too many objects, whereas high values could lead to empty results), a more effective control on the cardinality of the result can be achieved by just requesting the *k* "best/top matches", e.g.:

(Q7) *"Find the 3 paintings whose texture is most similar to a given texture vector"*.

FORMALIZATION

In this section we present the formal material underlying the SAME^{W} algebra. Our data model is a fuzzy extension of the relational model. A relation schema is formed by a relation name, *R*, and a subset of attributes, $X = \{A_1, \ldots, A_n\}$, where each A_i has a corresponding value domain, $\mathrm{dom}(A_i)$. For simplicity, we adopt the conventional list notation for sets, thus writing *A* for *{A}* and *XY* for $X \cup Y$. A tuple *t* over *R(X)* is any element of $\mathrm{dom}(X) = \mathrm{dom}(A_1) \times \ldots \times \mathrm{dom}(A_n)$, and $t.A_i$ denotes the value of A_i in *t*. For any $Y \subseteq X$, *t.Y* denotes the restriction of *t* on *Y*, that is, the (sub-)tuple obtained from *t* by considering only the values of the attributes in *Y*.

Given a tuple *t*, a fuzzy attribute *A* in *t* is formed by two components, A^{ν} (the "value") and A^{μ} (the "score") that, intuitively, have the following meaning: "*t* fits A^{ν} with score A^{μ}". A fuzzy relation *r* over the schema *R(X)* is a fuzzy set of tuples characterized by a membership function μ_R, which represents how much a given tuple "fits" the concept expressed by *R(X)*. In the following, the notation $t.\mu_R$ will be used with the same meaning of $\mu_R(t)$.

Predicates are combined into formulas according to the syntax $f ::= p \mid f \wedge f \mid f \vee f \mid \neg f \mid (f)$, where *f* is a formula and *p* is a predicate. The *evaluation* of *f* on a tuple *t* is a score, $s(f,t) \in [0,1]$, which says how much *t* satisfies *f*. How $s(f,t)$ depends on (the evaluation on *t* of) the predicates in *f* is intentionally left unspecified, in order to achieve parametricity with respect to the semantics of logical operators. More precisely, the score $s(f,t)$ is computed by means of a so-called "scoring function", s_f, whose arguments are the scores, $s(p_i,t)$, of tuple *t* with respect to the predicates appearing in *f*, that is, $s(f(p_1,\ldots,p_n),t) = s_f(s(p_1,t),\ldots,s(p_n,t))$. A *similarity predicate* has either the form $A \approx v$, where *A* is a feature attribute, $v \in \mathrm{dom}(A)$ is a constant, and $\approx$ is a *similarity operator*, or $A_1 \approx A_2$, where both A_1 and A_2 are over the same domain. The evaluation of *p* on *t* returns a score, $s(p,t) \in [0,1]$, which says

how much *t.A* is similar to the value *v*. The evaluation on *t* of a *fuzzy predicate q: A = w*, where *w* is a fuzzy set, is the score $s(q,t) = t.A^{\mu}$, if $t.A^{v} = w$, otherwise $s(q,t) = 0$. For fuzzy predicates of the form $q: A_1 = A_2$, it is $s(q,t) = 0$ if $t.A_1^{v} \neq t.A_2^{v}$, otherwise the score is computed as a "parametric conjunction" of the two membership degrees, that is, $s(q,t) = s_{\wedge}(t.A_1^{\mu}, t.A_2^{\mu})$, where $s_{\wedge}$ denotes the *And* scoring function. For the sake of definiteness, in the following we consider scoring functions corresponding to fuzzy *t-norms* and *t-conorms* (Klir, 1995), for which the *And* ($\wedge$) and *Or* ($\vee$) operators are both associative and commutative, and, together with the *Not* ($\neg$) operator, satisfy *boundary* and *monotonicity* conditions (Klir, 1995). For instance, FS (fuzzy standard) and FA (fuzzy algebraic) interpretations of the logical operators are as follows:

	FS	**FA**
$s(f_1 \wedge f_2, t)$	$\min(s(f_1,t), s(f_2,t))$	$s(f_1,t) \cdot s(f_2,t)$
$s(f_1 \vee f_2, t)$	$\max(s(f_1,t), s(f_2,t))$	$s(f_1,t) + s(f_2,t) - s(f_1,t) \cdot s(f_2,t)$
$s(\neg f, t)$	$1 - s(f,t)$	$1 - s(f,t)$

User preferences are expressed by means of *weights* as shown in (Fagin, 1997). The weighted version s_{f_Θ} of a scoring function s_f for a formula *f* on a set of predicates $p_1, \ldots, p_n$ is defined as follows. Let $x_i = s(p_i,t), \Theta = [\theta_1, \ldots, \theta_n]$, with $\theta_i \in [0,1]$ and $\sum_i \theta_i = 1$, and assume, without loss of generality, $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_n$. Then:

$$s_{f_\Theta}(x_1, \ldots, x_n) = (\theta_1 - \theta_2) \cdot x_1 + 2 \cdot (\theta_2 - \theta_3) \cdot s_f(x_1, x_2) + \ldots + n \cdot \theta_n \cdot s_f(x_1, \ldots, x_n)$$

Although above formula is usually used to weight the predicates appearing in a formula, our position is that *whenever scores have to be "combined", then a weighting should be allowed*. Accordingly, most of the SAME^{W} operators that compute new tuples' scores can use weights.

THE SAME^{W} ALGEBRA

Basic operators of SAME^{W} conservatively extend those of Relational Algebra (RA) in such a way that, if no "imprecision" is involved in the evaluation of an expression, the semantics of RA applies. Genericity with respect to different semantics is achieved by defining SAME^{W} operators in terms of the (generic) scoring functions of the logical operators. Thus, if a given semantics is adopted for formulas, the same is used by SAME^{W} operators. As an example, the semantics of Union ($\cup$) is based on that of the *Or* ($\vee$) operator.

In order to show the potentialities and flexibility of SAME^{W}, we refer to the *Virtual Museum* database in Figure 1. For lack of space, we only present a subset of SAME^{W} operators. A complete description, as well as the formal specification of operators' semantics, can be found in (Ciaccia, 2000-b). For

simplicity, in the following examples the computation of scores is always based on the FS language.

UffiziPaintings (UP)

PId	Title	Author	Room	Color	Texture
P015	Annunciazione	Leonardo	15	green: 0.78	T0015
P002	Adorazione dei Magi	Leonardo	15	red: 0.8	T0002
P005	Battesimo di Cristo	Leonardo	15	red: 0.5	T0005
P004	Madonna del Roseto	Botticelli	10	yellow: 0.6	T0004
P003	Madonna d'Ognissanti	Giotto	2	yellow: 0.85	T0003

LouvrePaintings (LP)

Pid	Title	Author	Color_Histogram	Texture
P007	La belle jardiniere	Raffaello	C007	T0007
P008	La joconde	Leonardo	C008	T0002
P011	Venus	Il Correggio	C011	T0011
P006	L'Apparition	A. Carracci	C006	T0004
P001	La Charité	A. del Sarto	C001	T0001

Conservation (C)

Pid	State	Museum
P002	good:0.58	Uffizi
P003	good: 0.8	Uffizi
P003	very good: 0.7	Uffizi
P004	good: 0.65	Uffizi
P004	very good: 0.5	Uffizi
P005	medium: 0.7	Uffizi
P015	very good: 0.78	Uffizi
P011	low: 0.35	Louvre
P007	good: 0.85	Louvre
P007	very good: 0.7	Louvre
P008	good: 0.67	Louvre
P006	low: 0.4	Louvre
P001	medium: 0.9	Louvre
P001	very good: 0.65	Louvre

PaintingClasses (PC)

Pid	Movement	μ
P002	Renaissance art	0.6
P003	Renaissance art	0.72
P004	Renaissance art	0.52
P005	Expressionist art	0.45
P015	Flemish art	0.38
P015	Cubism	0.2
P011	Expressionist art	0.25
P007	Impressionist art	0.84
P008	Renaissance art	0.9
P008	Cubism	0.1
P006	Impressionist art	0.52
P001	Flemish art	0.31
P001	Renaissance art	0.75

Figure 1. The Virtual Museum database.

Selection (σ). The Selection operator applies a formula *f* to the tuples of a relation *r* and filters out those that do not satisfy *f*. The novel point here is that, as an effect of *f* and of weights, *the score of a tuple t can change*. Weights can be used for two complementary needs: in the first case, they weight the importance of predicates in *f*, whereas in the second they are used to perform a *weighted conjunction*, between the score computed by *f* and the "input" tuple score, $t.\mu_R$.

Example 1. Query (Q3) can be expressed in SAME^{W} as:

$\sigma_{\text{Color}=`red' \wedge \text{Texture} \approx t}$(UffiziPaintings)

where *t* stands for the input texture vector. Assuming that the following scores are obtained for the texture predicate:

Texture	**Score**
T0001	0.85
T0002	0.58
T0003	0.45
T0004	0.8
T0005	0.6
T0007	0.7
T0011	0.55
T0015	0.72

the resulting tuples are:

UffiziPaintings

Pid	**Title**	**Author**	**Room**	**Color**	**Texture**	μ
P002	Adorazione dei Magi	Leonardo	15	red: 0.8	T0002	0.58
P005	Battesimo di Cristo	Leonardo	15	red: 0.5	T0005	0.5

Now consider the presence of weights. Query (Q4) can be expressed as:

$\sigma_{\text{Color}=`red'^{2\ 3} \wedge \text{Texture} \approx t^{1\ 3}}$(UffiziPaintings)

and the result becomes:

UffiziPaintings

Pid	**Title**	**Author**	**Room**	**Color**	**Texture**	μ
P002	Adorazione dei Magi	Leonardo	15	red: 0.8	T0002	0.653
P005	Battesimo di Cristo	Leonardo	15	red: 0.5	T0005	0.5

Projection (π). As in RA, the Projection operator removes a set of attributes and then eliminates duplicate tuples. Projection can also be used to *discard* scores, both of fuzzy attributes and of the whole tuple. In this case, however, in order to guarantee consistency of subsequent operations, such scores are simply set to 1, so that they can still be referenced in the resulting schema. This captures the intuition that if we discard, say, the tuples' scores, then the result is a crisp relation, that is, a fuzzy relation whose tuples all have score 1. In order to retain the original tuples' scores, "μ" has to be specified in the Projection list of attributes, whereas if A_i is a fuzzy attribute, specifying A_i^v in such list sets to 1 the scores for A_i. Note that, when tuples' scores are preserved, the scores of

result tuples are computed by considering the "parametric disjunction", $s_{\vee}$, of the scores of all duplicate tuples arising from the Projection.

Example 2. Consider the query which returns all the artistic movements represented in the Virtual Museum, together with corresponding classification scores. This can be expressed as:

$$\pi_{\text{Movement},\mu}(\text{PaintingClasses})$$

and the selected tuples are:

Movement	μ
Renaissance art	0.9
Expressionist art	0.45
Flemish art	0.38
Cubism	0.2
Impressionist art	0.84

Join («). In SAME^{W} the weighted (natural) Join is an *n*-ary operator that, given *n* relations r_i with schemas $R_i(X_i)$, computes the score of a tuple *t* as a *weighted conjunction*, $s_{\wedge}^{\Theta}(t_1.\mu_{R_1},\ldots,t_n.\mu_{Rn})$ with $\Theta=[\theta_1,\ldots,\theta_n]$, of the scores of matching tuples. For simplicity, we also use the infix notation, $R_1 \ll^{[\theta_1,\theta_2]} R_2$, when only two operands are present.

Example 3. The following query looks for yellow (with weight 0.7) paintings in the Uffizi Gallery and in a very good state of conservation (with weight 0.3):

$$\sigma_{\text{Color}=`yellow'}(\text{UffiziPaintings}) \ll^{[0.7,0.3]} \sigma_{\text{State}=`very\ \ good'}(\text{Conservation})$$

The resulting tuples are the following ones, where, for space reasons, some columns are omitted:

Pid	**Title**	**Color**	**State**	**Museum**	μ
P004	M. del Roseto	yellow: 0.6	very good: 0.5	Uffizi	0.54
P003	M. d'Ognissanti	yellow: 0.85	very good:0.7	Uffizi	0.76

In order to see how final scores are determined, consider tuple P003. Since the predicate on color yields a score 0.85, whereas the predicate on state of conservation is satisfied with score 0.7, the weighted conjunction leads to:

$$\mu = (0.7-0.3)\cdot 0.85 + 2\cdot 0.3\cdot \min(0.85, 0.7) = 0.76 \ .$$

Union ($\cup$). Also the Union is an *n*-ary operator, where the score of a result tuple *t* is a *weighted disjunction*, $s_{\vee}(t.\mu_{R_1},\ldots,t.\mu_{R_n})$, of the input tuples' scores. Note that, because of the presence of weights, Union is not associative

anymore. This implies that the *n*-ary Union cannot be defined in terms of *n-1* binary unions, as it happens in RA. As with Join, infix notation is used for a binary Union.

Example 4. The following SAME^{W} expression retrieves texture vectors and scores of those paintings that belong to the Renaissance movement. Further, it specifies that paintings from Uffizi (UP) have relevance 0.6 and, consequently, those from Louvre (LP) have relevance 0.4.

$$\pi_{\text{Texture},\mu}\left(\text{UP} \ll \sigma_{\text{Movement}=`\textit{Renaissance art}'}\left(\text{PaintingClasses}\right)\right) \cup^{[0.6,0.4]} \pi_{\text{Texture},\mu}\left(\text{LP} \ll \sigma_{\text{Movement}=`\textit{Renaissance art}'}\left(\text{PaintingClasses}\right)\right)$$

Then, the resulting tuples are:

Texture	μ
T0001	0.6
T0002	0.84
T0003	0.72
T0004	0.52

Top (τ). The Top operator retrieves the first *k* (*k* is an input parameter) "best ranked" tuples of a relation *r*, where the ranking criterion is expressed by a ranking function *g*. If weights are used to rank tuples according to g_{Θ}, then *g* has to be a formula of predicates. If *r* has no more than *k* tuples, then Top has no effect. When *g* is omitted, the *default* ranking criterion, based on the score of tuples, applies, thus the *k* tuples with the highest scores are returned. In all cases, ties are arbitrarily broken.

Example 5. The following expression returns the two paintings in the Uffizi Gallery whose texture vector is most similar to a given one. Since the ranking criterion of the Top operator is not specified, tuples are retrieved according to their scores, as computed by the Selection operator.

$$\tau^{2}\left(\sigma_{\text{Texture}\approx t}\left(\text{UffiziPaintings}\right)\right)$$

Given the similarity table of Example 1 the result is:

Pid	**Title**	**Author**	**Room**	**Color**	**Texture**	μ
P015	Annunciazione	Leonardo	15	green: 0.78	T0015	0.72
P004	M. del Roseto	Botticelli	10	yellow: 0.6	T0004	0.8

Cut (γ). The Cut operator simply "cuts off" those tuples that do not satisfy a formula *g*. Unlike Selection, Cut *does not change tuples' scores.* The major reason to introduce Cut is the need of expressing *(threshold) conditions on*

tuples' scores, e.g. $\mu > 0.6$. Such predicates cannot be part of a Selection, since they do not commute with others (Ciaccia, 2000-b).

Example 6. The following expression retrieves only the paintings in the Uffizi Gallery whose texture vector is similar to the given value *t* with a score higher than 0.75:

$$\gamma_{\mu>0.75}\left(\sigma_{\text{Texture}\approx t}(\text{UffiziPaintings})\right)$$

As a result of the threshold condition, only one tuple is returned:

Pid	**Title**	**Author**	**Room**	**Color**	**Texture**	μ
P004	M. del Roseto	Botticelli	10	yellow: 0.6	T0004	0.8

QUERY OPTIMIZATION

The introduction of fuzzy attributes and fuzzy relations, as well as the presence of weights in user queries, introduces new aspects to be considered when reasoning about query rewriting and evaluation. This is mainly due to the different semantics all the operators have with respect to the Relational Algebra, as well as to the introduction of new operators.

To give a flavor of the optimization opportunities that are available in SAME^{W}, in the following we present an example of query rewriting that exploits the following equivalence rules, whose proofs can be found in (Ciaccia, 2000-b).

$$\gamma_{\alpha}\left(\sigma_{p_1^{\theta_1}\wedge p_2^{\theta_2}}(E)\right) \equiv \gamma_{\alpha}\left(\sigma_{p_2}^{[\theta_2,\theta_1]}\left(\gamma_{\alpha}\left(\sigma_{p_1}(E)\right)\right)\right) \qquad \textbf{(R.1)}$$

$$\sigma_{p_1\wedge\ldots\wedge p_n}(\ll(E_1,\ldots,E_n)) \equiv \ll(\sigma_{p_1}(E_1),\ldots,\sigma_{p_n}(E_n)) \qquad \textbf{(R.2)}$$

$$\gamma_{\alpha}(\ll^{[\theta_1,\ldots,\theta_n]}(E_1,\ldots,E_n)) \equiv \gamma_{\alpha}(\ll^{[\theta_1,\ldots,\theta_n]}(\gamma_{\alpha_1}(E_1),\ldots,\gamma_{\alpha_n}(E_n))) \qquad \textbf{(R.3)}$$

where R.1 applies when $\theta_1 \geq \theta_2$, and in R.3 it is:

$$\alpha_i = \left(\alpha - \sum_{j=1}^{i-1} j\cdot(\theta_j - \theta_{j+1})\right) \left(1 - \sum_{j=1}^{i-1} j\cdot(\theta_j - \theta_{j+1})\right) \qquad i \in [1,\ldots,n]$$

Briefly, rule R.1 allows the predicate with the highest weight to be "pushed-down" and a Cut to be added to discard tuples; rule, R.2 distributes predicates of a conjunctive formula over the corresponding Join operands; and rule R.3 allows the introduction of new Cut operators over the operands of a weighted Join, with thresholds values determined by the set of weights, as shown above.

The example query we consider requests information on paintings in the Uffizi Gallery, whose painter is Leonardo, having a good state of conservation (weight 0.6), and whose texture is similar to a given value *t* (weight 0.4). The resulting

score has to be greater than 0.7. It is anticipated that the evaluation of the predicate on Texture is costly, thus query optimization should aim to evaluate the predicate on as few tuples as possible. The initial query formulation is:

$$\gamma_{0.7}\left(\sigma_{\text{State}=`good',0.6 \wedge \text{Texture}\approx t}{}^{0.4}\left(\sigma_{\text{Author}='Leonardo'}(\text{C} \ll \text{UP})\right)\right)$$

where C stands for Conservation and UP for UffiziPaintings. Thanks to rule R.1, the predicate on State can be pushed-down, i.e.:

$$\gamma_{0.7}\left(\sigma^{[0.4,0.6]}_{\text{Texture}\approx t}\left(\gamma_{0.7}\left(\sigma_{\text{State}=`good'}\left(\sigma_{\text{Author}=`Leonardo'}(\text{C} \ll \text{UP})\right)\right)\right)\right)$$

Then, from the associativity of t-norms and the definition of Selection:

$$\gamma_{0.7}\left(\sigma^{[0.4,0.6]}_{\text{Texture}\approx t}\left(\gamma_{0.7}\left(\sigma_{\text{State}=`good' \wedge \text{Author}=`Leonardo'}(\text{C} \ll \text{UP})\right)\right)\right)$$

Now, the application of rule R.2 leads to:

$$\gamma_{0.7}\left(\sigma^{[0.4,0.6]}_{\text{Texture}\approx t}\left(\gamma_{0.7}\left(\sigma_{\text{State}=`good'}(\text{C}) \ll \sigma_{\text{Author}=`Leonardo'}(\text{UP})\right)\right)\right)$$

At this point rule R.3 can be applied to the inner Cut, which yields:

$$\gamma_{0.7}\left(\sigma^{[0.4,0.6]}_{\text{Texture}\approx t}\left(\gamma_{0.7}\left(\gamma_{0.7}\left(\sigma_{\text{State}=`good'}(C)\right) \ll \gamma_{0.7}\left(\sigma_{\text{Author}=`Leonardo'}(\text{UP})\right)\right)\right)\right)$$

The Cut on UP can be eliminated, since the relation is crisp and the Selection has a Boolean predicate, thus scores of tuples are equal to 1. Since tuples in the left Join operand are guaranteed to have score higher than 0.7 (due to the Cut on C), whereas tuples in the right operand are crisp, the resulting scores of the joined tuples are also higher than 0.7. This means that the Cut following the Join can be safely dropped:

$$\gamma_{0.7}\left(\sigma^{[0.4,0.6]}_{\text{Texture}\approx t}\left(\gamma_{0.7}\left(\sigma_{\text{State}=`good'}(\text{C})\right) \ll \sigma_{\text{Author}=`Leonardo'}(\text{UP})\right)\right)$$

In conclusion, the costly-to-evaluate predicate on Texture is applied only to those tuples that already satisfy the other (cheap) predicates, thus leading to considerable cost reduction.

CONCLUSIONS AND FUTURE WORK

We have introduced an extended relational algebra, called SAME^{W}, suitable for expressing complex similarity queries on multimedia data, and dealing with imprecision and user preferences in a uniform way. The novel query features include new operators like the Cut and the Top, useful to express “threshold” and “best-matches” queries, the presence of similarity and fuzzy predicates, the use of scores to rank the answers, and the possibility of weighting subqueries and predicates in order to express their relevance in user preferences. A working example, dealing with collections of paintings retrieved from the Uffizi and Louvre Web sites, has been used in order to describe the behavior of the

SAME^W operators and to sketch aspects related to query rewriting and optimization, a subject that, for lack of space, we have not covered here in its full extension.

The optimization issues arising in the context of the SAME^W algebra are the main line of research we are following. Such issues do not restrict only to the study of equivalence rules among SAME^W expressions including new operators and weights. Indeed, the presence of similarity predicates influences heavily the definition of cost models, since this kind of queries can be expensive to compute, thus making the "selection push-down" rule of thumb not valid anymore. Along this line, a prototype implementation of the SAME^W algebra is under development.

In addition to optimization, there are many other issues worth investigating. Among them, we mention the development of complex data models, in order to better modeling complex multimedia objects. In this context, it is also of particular relevance the development of an intuitive, yet fully expressive, non-procedural user query language, along the lines of the SQL standard. Also, the SAME^W algebra can be modified for modeling and querying semistructured data like those expressed in XML. A further interesting research issue is related to relevance feedback. The fact that the result of a query does not match user expectation is common in multimedia systems, and leads to a "closed-loop" interactive process, where user evaluation is fed back to the query engine and then taken into account to compute a "better" result, and so on. The proposed algebra can be extended to deal with user judgements that could be used to adjust weights for query reformulation.

REFERENCES

(Adali, 1998) S. Adali, P. Bonatti, M.L. Sapino, and V.S. Subrahmanian. A Multi-Similarity Algebra. In *Proc. of the 1998 ACM-SIGMOD Int. Conf. on Management of Data*, pp. 402--413, Seattle, WA, 1998.

(Carey, 1997) M.J. Carey and D. Kossmann. On Saying "Enough Already!" in SQL. In *Proc.of the 1997 ACM SIGMOD Int. Conf. on Management of Data*, pp. 219--230, Tucson, AZ, 1997.

(Ciaccia, 1997) P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proc.of the 23rd VLDB Int. Conf.*, pp. 426--435, Athens, Greece, 1997.

(Ciaccia, 1998) P. Ciaccia, M. Patella, and P. Zezula. Processing Complex Similarity Queries with Distance-based Access Methods. In *Proc. of the 6th Int. Conf. on Extending Database Technology (EDBT'98)*, pp. 9--23, Valencia, Spain, 1998.

(Ciaccia, 2000-a) P. Ciaccia and M. Patella. PAC Nearest Neighbor Queries: Approximate and Controlled Search in High-Dimensional and Metric Spaces. In *Proc.of the 16th Int. Conf. on Data Engineering (ICDE 2000)*, San Diego, CA, 2000.

(Ciaccia, 2000-b) P. Ciaccia, D. Montesi, W. Penzo, and A. Trombetta. Imprecision and User Preferences in Multimedia Queries: A Generic Algebraic Approach. In *First Int. Symp. on Foundations of Information and Knowledge Systems, FoIKS 2000*, pp. 50--71, Burg, Germany, 2000 (also available at ftp://ftp-db.deis.unibo.it/pub/paolo/FOIKS00/FOIKS00.ps.gz).

(Fagin, 1996) R. Fagin. Combining Fuzzy Information from Multiple Systems. In *Proc. of the 15th ACM Symposium on Principles of Database Systems (PODS'96)*, pp. 216--226, Montreal, Canada, 1996.

(Fagin, 1997) R. Fagin and E.L. Wimmers. Incorporating User Preferences in Multimedia Queries. In *Proc. of the 6th ICDT Int. Conf.*, pp. 247--261, Delphi, Greece, 1997.

(Flickner, 1995) M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9):23--32, 1995.

(Gravano, 1997) L. Gravano and H. Garcia-Molina. Merging Ranks from Heterogeneous Internet Sources, In *Proc. of the 23rd VLDB Int. Conf.*, pp. 196--205, Athens, Greece, 1997.

(Klir, 1995) G.J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic*. Prentice Hall PTR, 1995.

(Lakshmanan, 1997) L.V.S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. *ACM Transactions on Database Systems*, 22(3), pp. 419--469, 1997.

(Montesi, 1999) D. Montesi and A. Trombetta. Similarity Search through Fuzzy Relational Algebra. In *Proc. of the 1st Int. Workshop on Similarity Search (IWOSS'99)*, Florence, Italy, 1999.

(Raju, 1988) K. Raju and A. Majumdar. Fuzzy Functional Dependencies and Lossless Join Decomposition of Fuzzy Relational Database Systems. *ACM Trans. on Database Systems*, 13(32):129--166, 1988.

(Seidl, 1997) T. Seidl and H.-P. Kriegel. Efficient User-Adaptable Similarity Search in Large Multimedia Databases.In *Proc. of the 23rd VLDB Int. Conf.*, pp. 506--515, Athens, Greece, 1997.

(Shivakumar, 1998) N. Shivakumar, H. Garcia-Molina, and C.S. Chekuri. Filtering with Approximate Predicates. In *Proc. of the 24th VLDB Int. Conf.*, pp. 263--274, New York, NY, 1998.

(Soffer, 1998) A. Soffer and H. Samet. Integrating Symbolic Images into a Multimedia Database System using Classification and Abstraction Approaches. *The VLDB Journal*, 7(4):253--274, 1998.

(Subrahmanian, 1998) V.S. Subrahmanian. *Principles of Multimedia Database Systems*. Morgan Kaufmann, 1998.