http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 8 (4): 411-426, 2009
ISSN 1812-5638

UMQA: An Internal Algebra for Querying Multimedia Contents

Zongda Wu, Zhongsheng Cao and Yuanzhen Wang
Institute of Database and Multimedia Technology, College of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan 430074, Poeple's Republic of China

Abstract: For internal query processing, we in this study discuss an operator-based algebraic language called UMQA, whose operators are formally similar to those of the relational algebraic system. To deal with UMQL's extensions for structure, feature and spatio-temporal queries, UMQA is also introduced with some new operators: structure selection (σ^{SE}), structure expansion (η), feature selection (σ^{FE}) and spatio-temporal selection (σ^{SP}), which make UMQA of equivalent capability with UMQL on multimedia query specification, but more suitable for internal query processing due to it representing multimedia queries in an algebraic way. We also introduce an approach to translate UMQL queries into UMQA plans equivalently and a powerful set of algebraic translation formulas that is important for query optimization by algebraic rewriting. Last, we summarize a UMQL prototype information system which uses UMQA as its internal processing algebra and briefly discuss the efficient implementation of UMQA operators.

Key words: Multimedia content, multimedia query language, internal algebra, relational algebra

INTRODUCTION

In the past decade, with the development of the internet and the availability of multimedia digital capturing devices such as image scanners, digital cameras and digital video cameras, the size of digital multimedia collection is increasing rapidly. Consequently, effective multimedia information retrieval and management techniques become more and more important. Multimedia content information represents what people sense when looking or listening, namely what are included by multimedia data and what features they behave themselves with, so it is more comprehensible for users and very important for content-based multimedia information retrieval. In recent years, many methods on extracting content information from multimedia original data have been proposed (Liu *et al*., 2007; Christel and Hauptmann, 2005). Presently, some high-level content information still can not be extracted effectively, but we believe that in the future more effective content extraction techniques will be proposed and more abundant multimedia contents will be extracted. Therefore, there should be a strong need to store, query and play multimedia content information from multimedia databases.

A multimedia query language is a useful facility to specify users' multimedia query requirements and therefore is one of the most essential components in a multimedia database system. Hence, for querying multimedia contents effectively, a powerful and friendly query language must be supplied first for users. Although traditional database query languages (e.g., SQL, OQL, etc.) have acquired great success, they do not suit uniform multimedia information retrieval, because the complex spatial and temporal relationships inherent in the wide range of multimedia data types make a multimedia query language different from its counterparts in traditional database management systems. In recent years, there have been many multimedia query language proposals (Tian *et al*., 1999; Balkir *et al*., 2002; Li and Ozsoyoglu, 1996; Lee *et al*., 1999a, b, 2000), most of which are either designed for one particular medium (e.g., images), or specialized for a particular application (e.g., digital libraries), consequently also not competent for uniform multimedia information retrieval.

In an earlier study, (Cao *et al*., 2007) of this project group, we have given a semi-structured data organization model and discussed a general-purpose multimedia query language called UMQL, which allows users to query various multimedia data uniformly based on their content information such as structure, feature, spatial relationship and temporal relationship. Later, to supply a friendlier interface for users, we designed and implemented a graphical environment (Wu *et al*., 2008), which uses UMQL as its internal language and to check the correctness for any UMQL query given by users and we proposed a grammar analysis model and implemented its corresponding grammar analyzer (Cao *et al*., 2008;

Corresponding Author: Zhongsheng Cao, Institute of Database and Multimedia Technology,
College of Computer Science and Technology, Huazhong University of Science and Technology,
Wuhan 430074, Poeple's Republic of China

Huang, 2008). However, UMQL is a textual declarative query language just as SQL or OQL, designed only for users to use and therefore not suitable for internal implementation, so it should be converted into some internal representations for being optimized and implemented efficiently. Internal query algebras are just such internal representations. Presently, however, most of existing query algebras are proposed for traditional relational database languages, object-oriented database languages (Shaw and Zdonik, 1990; Sophie and Claude, 1992) or XML document query languages (Meng *et al*., 2006). It is unfortunate that, although these algebras have acquired well application effects, they don't suit for multimedia query languages, due to multimedia query different particularities.

In this study, we present an operator-based algebraic language called UMQA, to process UMQL queries, especially to establish foundation for query optimization and implementation so, which is an internal algebra, designed specially for internal implementation, i.e., not for users to use directly. UMQA is similar to the traditional relational algebra, both of formally similar algebraic operations. To support various requirements for querying on multimedia contents, UMQL introduces new conditional expressions into its WHERE clause, mainly including structure condition, feature condition and spatio-temporal condition. Accordingly, respectively corresponding to each of them, UMQA is also extended with some new operators, i.e., structure selection (σ^{SE}), feature selection (σ^{FE}) and spatio-temporal selection (σ^{SP}). Moreover, structure expansion (η) is introduced, used for expanding content objects to obtain all their child objects based on a path expression and the original selection operator of the relational algebraic system is inherited and called normal selection (σ^{NL}). All these make UMQA of equivalent capability with UMQL on multimedia query specification. We then present an approach to transform UMQL queries into their equivalent UMQA plans.

Because of the extensions for new operators, for UMQA, a majority of traditional equivalent translation formulas for the relational algebraic system are obviously no longer applicable. So, in this study, we also present a powerful set of equivalent algebraic translation formulas, specialized for UMQA, mostly concentrating on the conjunction laws and exchange laws among algebraic operators, which is very important for query optimization having a small algebraic rewriting search space and consequently establishes the foundation for internal query optimization.

UMQL AND ITS DATA MODEL

Data organization model: For representing and modeling multimedia data and their corresponding content information, many logic data models have been proposed (Zuo and Cao, 2008; Wang *et al*., 2007; Aygun and Yazici, 2004), including the categories of annotation-based, rich semantic etc. We here no longer discuss multimedia modeling approach and assume that a variety of multimedia contents have been extracted and stored into multimedia databases but organized in a semi-structured form, i.e., we use a semi-structured data organization model to organize and store multimedia data and their content information, acting as query operating foundation. There are other data organization forms such as the relational model and object-oriented one, but we believe that, semi-structured organization forms are more suitable for storing complex multimedia contents, because most of existing multimedia modeling methods, such as G3M (Zuo and Cao, 2008) and MPEG7, also organize their modeling information in semi-structured form.

The data origination model that UMQL is based on includes such basic notions as constructed data type, collection data type, object, child object and so on. We now briefly describe these notions. A constructed data type is a composite data structure comprising of predefined data types (such as FLOAT, INTEGER etc.), collection data types and other constructed data types, whose instance is an object. One instance of any collection data type is a composite value comprising of zero or more elements, all of the same data type. Every basic item of an object is an attribute, whose value is called the attribute value of the object and also called the child objects of the object if the data type of the item is a collection data type or a constructed data type. To simplify presentation, a constructed data type represents both the data type name and the structure of objects belonging to the data type.

In Fig. 1a, we give a simplified data organization schema, where INTEGER, CHAR, STRING, DATE and OBJECT are all predefined data types and PERSON, MOVIE, VIDEO, CLIP and EVENT are all constructed data types. The data schema denotes that each video consists of a sequence of video clips and video events; each video clip contains several salient content objects and some relationships among these salient objects and the data types of salient content object are predefined by system. In Fig. 1b, we also present a simple instantiation of the data schema but some attributes not concerned by the examples in the subsequent sections are omitted for simplicity.

Multimedia query language: This section by a simplified query example summarizes those features of UMQL relevant to the use and querying of multimedia content information, however, whose more particular syntax and semantic features are given in present studies (Cao *et al*., 2007, 2008; Wu *et al*., 2008; Huang, 2008). UMQL uses a variable to represent a group of content objects of the

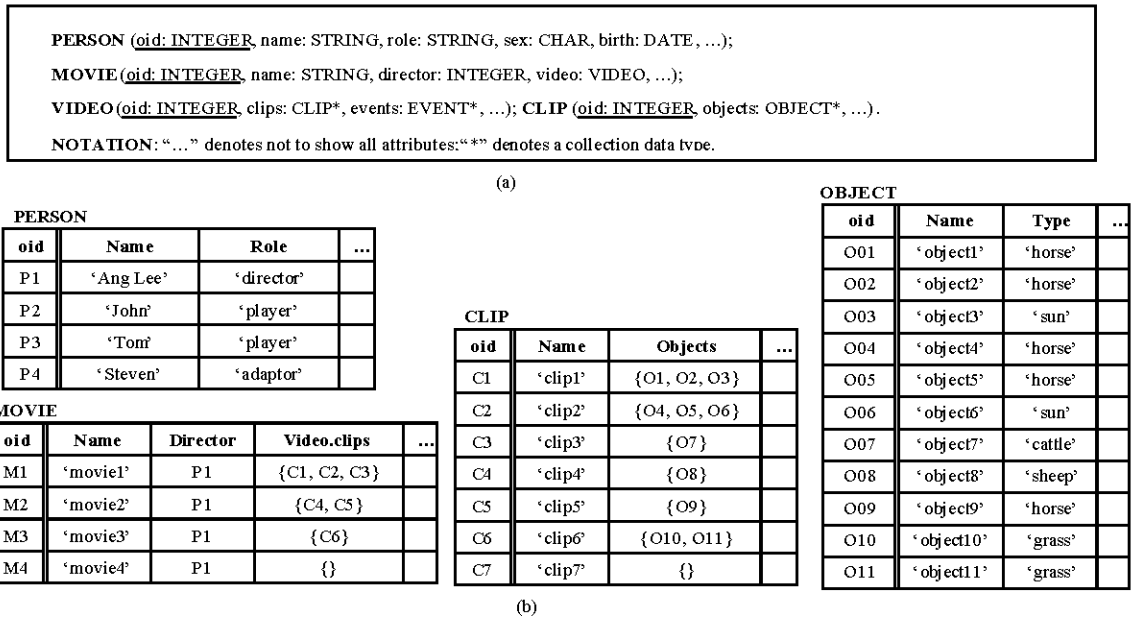

Fig. 1: A data organization schema and its corresponding instantiation

same data type, all satisfying the same conditional restriction. A UMQL query is represented by one or more basic statements, i.e., one external query and some internal queries. As in SQL, UMQL uses the traditional SELECT-FROM-WHERE statement for querying, but it extends the WHERE clause with new structure condition, feature condition and spatio-temporal condition, where the FROM clause is to declare some top-level variables; the structure conditional expression is also to declare variables, but at the same time to describe the ancestor-child relationships among these variables; the spatio-temporal conditional expression is to describe the spatial and temporal relationships among variables and the normal or feature conditional expression is to specify the common normal or feature restriction for the same group of content objects in each variable.

Example 1: Retrieve movies directed by Ang Lee and satisfying the conditions listed below

- In each of the movies, there are a video clip that contains more than fifty five video frames
- And, in the video clip, there are three salient content objects, two 'horse', whose color features are both similar (similarity more than 0.75) to that of the given image 'horse.bmp' and one 'sun', whose shape feature is similar to that of the given image, 'sun.bmp'
- Last, for the three content objects, the 'sun' is located on the top of the two 'horse'. The query is specified by UMQL as follows:

SELECT m.name **FROM** MOVIE m, PERSON d
(variable declarations)

WHERE clip **IN** m.video.clips **AND** horse(2), sun(1) **IN** clip.objects
(structure conditional expression)

AND d.id = m.director **AND** d.name = 'Ang Lee' **AND** d.role = 'director'
(normal and join conditional expression)

AND frame(clip) > 55 **AND** is(horse, 'horse') **AND** is(sun, 'sun')
(feature conditional expression)

AND color(horse, 'horse.bmp') > 0.75 **AND** shape(sun, 'sun.bmp') > 0.75
(feature conditional expression)

AND horse **BEFORE**[Y] sun
(spatio-temporal conditional expression)

In example 1, we first in the structure conditional expression specify the structure framework of target multimedia data by declaring some variables (i.e., m, clip, horse and sun) and the ancestor-child relationships among them; then the other conditional expressions are employed to describe the common normal, feature and spatio-temporal restrictions for each group of content objects represented by each of these variables.

From example 1, we know that users specify UMQL queries using the interpretation semantics, which means that for each instance of query operating objects in multimedia databases, the methods referred to within the UMQL query statements are evaluated and the conditional restrictions are checked. The outputs are then retrieved if all the restrictions in the query statements are satisfied. For UMQL, compared with SQL, there are three new conditional expressions, i.e., structure, feature and spatio-temporal conditional expressions.

Structure conditional expression: A structure conditional expression is employed to support querying based on the structure of multimedia content information, whose basic conditional item is generally described as $F = d_1(a_1), d_2(a_2), \ldots, d_n(a_n)$ IN $d.s_1.s_2.\ldots.s_m$, where, d_i (for $i = 1, 2, \ldots, n$, $n \geq 1$) is a new variable declared in F, a_i is a positive integer called the correlative number of d_i and $d.s_1.s_2.\ldots.s_m$ ($m \geq 1$; d is also a variable called the immediate ancestor variable of d_1, d_2, ... and d_n and $d.s_1.s_2.\ldots.s_{i+1}$ is an attribute of $d.s_1.s_2.\ldots.s_i$ for $i = 1, 2, \ldots, m-1$) is a path expression. If the conditional restrictions on d_1, d_2, ... and d_n are C_1, C_2, ... and C_n respectively in the WHERE clause, then any content object o in the variable d satisfies the conditional item F, if and only if there exist $(a_1+a_2+\ldots+a_n)$ child objects expanded from o based on the path $d.s_1.s_2.\ldots.s_m$, where a_1 objects satisfy C_1, a_2 objects of the rest satisfy C_2 and so on.

Feature conditional expression: A feature conditional expression is employed to support querying based on bottom-level features, semantic notions or other multimedia features, whose implementations depend on functions supplied by system or users. It is allowable for users to define themselves feature functions, as long as abiding by the restrictive rules on function definition.

Spatio-temporal conditional: A "spatio-temporal conditional expression" is employed to support querying based on the spatio-temporal information widely inherent in various multimedia salient content objects. UMQL implements all temporal relations as temporal operators defined by Allen's temporal interval algebra (Allen, 1983) i.e., BEFORE, FINISHES, OVERLAPS, MEETS, STARTS, DURING and EQUALS. However, some different with Allen's temporal algebra, UMQL uses these relations to implement spatio-temporal queries uniformly, so we call them spatio-temporal operators. We use the projected (on X, Y and Z axes) interval relationships and the information about their overlap to represent the spatial relationships among spatial objects. Therefore, the spatial relationships can be represented by the seven interval relations uniformly. The spatio-temporal operators are the construct foundation of a spatio-temporal conditional expression.

UMQA: AN INTERNAL ALGEBRAIC LANGUAGE

UMQL is a powerful general-purpose multimedia query language that allows users to query a variety of multimedia data uniformly based on content information such as structure, feature, spatial relationship and temporal relationship, but it is declarative, designed for users to use and not suitable for internal implementation, so it should be converted into some internal representations for being optimized and implemented efficiently. Internal algebras are such internal representations; however, all present existing query algebras (Shaw and Zdonik, 1990; Tian *et al*., 1999) are designed for themselves applications and obviously not applicable. In this section, we introduce an operator-based algebraic language called UMQA, for internally representing and processing UMQL queries, which is extended with some new operators for those conditional expressions new introduced into UMQL, on the basis of the traditional relational algebraic system.

Query generation graph: To internally represent many collections of content objects obtained from multimedia databases, UMQA uses a query generation graph, each vertex representing a variable bound with a collection of content objects and each edge that connects two vertices representing the ancestor-child relationship between the variables denoted by the vertices. Below we elaborate on the organization structure of a query generation graph.

Definition 1: A query generation graph G = (V, E) consists of V(G), a nonempty set of vertices and E(G), a set of ordered pairs of distinct elements of V(G) called edges and:

- Each edge (u, v) of E(G) represents an ancestor-child relationship between two variables denoted by the vertices u and v and have $\forall(u,v)\forall(u',v')(((u,v)\in E(G),(u',v')\in E(G),u=u')\rightarrow(v=v'))$
- Each vertex u of V(G) represents a variable comprising of a three-tuple form (N, O, R), where, u [N] denotes the name of the variable, u [O] is a set of objects of the same data type bound to the variable and u [R] employed to point out the immediate ancestor object for each object in u [O], is a set of ordered pairs of oids, satisfying,

$$\begin{cases}\forall(x,y)((x,y)\in u[R]\rightarrow x\in \&u[O], y\in \&v[O]) & \text{if } \exists v(v\in V(G),(u,v)\in E(G))\\ \forall(x,y)((x,y)\in u[R]\rightarrow x\in \&u[O], y=0) & \text{otherwise}\end{cases}$$

where, & a represents the oid of a if a is an object, or else represents a collection of oids, each denoting an object in a, if a is a collection of content objects.

UMQA operands are all query generation graph. As implementing a UMQA plan, all the operators act on a query generation graph, in which, scan and structure expansion are to fetch content information from multimedia databases and to construct and initialize a query generation graph for storing the information; then normal selection, structure selection, feature selection and spatio-temporal selection are to remove the content objects stored in the vertex variables but not satisfying the given conditional restrictions.

UMQA algebraic operators: The extensions for UMQA mainly concentrate on selection operators, including normal selection, structure selection, feature selection and spatio-temporal selection, respectively corresponding with normal condition, structure condition, feature condition and spatio-temporal condition. Moreover, structure expansion is introduced to expand content objects to obtain their entire child content objects based on a path expression.

Data extraction operators: Scan and structure expansion: Scan (ε) and structure expansion (η) are both data extraction operators, designed for obtaining content information from multimedia databases, then constructing and initializing a query generation graph to store these contents, acting as the input operands of the other subsequent operators in the same plan.

Definition 2: Given a basic scan conditional item "d←D", where, "d" and "D" are both character strings and "D" is the name of a constructed data type, representing a collection of content objects in multimedia database, the scan operator on the conditional item, ε["d←D"](Ø), constructs and outputs a query generation graph G_O as follows:

$$V(G_O) = \{v : (N : "d", O : \{o \mid o \in D\}, R : \{(x, y) \mid x \in \&O, y = 0\})\}; E(G_O) = \varnothing$$

This means that, for the scan operator, its input operand is empty and its output is the query generation graph comprising of only one vertex v named by the character string "d" and all its content objects obtained from the objects' collection "D" ($o \in D$), each without any ancestor ($x \in \& O, y = 0$).

For a structure expansion operator, its operating expression is a structure conditional item. It is employed to expand each object in a variable of its input query generation graph for obtaining all child objects; bind all these child objects to each new variable declared in its input conditional item and put these new variables into the input graph to output a new query generation graph. However, in this process, an expansion function employed to expand objects is essential.

Definition 3: Let P and O, respectively are the domain of path expressions and the domain of content objects. If $\varnothing \subset D \subseteq O \times P$, then the mapping f: $D \to 2^0$, is an expansion function and for $o \in O$ and $p \in P$, f (o, p) is a collection comprising of all the content objects that the object o can reach along the path p.

Definition 4: Given a basic structure conditional item $d_1(a_1), d_2(a_2), \ldots, d_n(a_n)$ IN $d.s_1.s_2. \ldots s_m$ ($n \geq 1$, $m \geq 1$), the structure expansion operator on the conditional item η ["$d_1(a_1), d_2(a_2), \ldots, d_n(a_n)$ IN $d.s_1.s_2. \ldots s_m$"](G_{in}), based on its input operand G_{in}, outputs a new query generation graph G_O as follows:

$$V(G_O) = V(G_{in}) \cup \{v_1 : (N : "d_1", O : \rho, R : \mu), v_2 : (N : "d_2", O : \rho, R : \mu), \ldots, v_n : (N : "d_n", O : \rho, R : \mu)\};$$
$$E(G_O) = E(G_{in}) \cup \{(u, v) \mid u \in V(G_O) - V(G_{in}), v[N] = "d"\}$$

This means that, the structure expansion operator outputs a new query generation graph, where n new vertices are produced, all of the same objects' collection O (all ρ) and the same R (all μ) and their ancestor variable is that of the vertex named by "d" ($u \in V(G_O) - V(G_{in})$, $v[N] =$ "d"). And:

$$\rho = \{o \mid \exists u \exists a (u \in V(G_{in}), u[N] = "d", a \in u[O], |f(a, "d.s_1.s_2.....s_m")| \geq (a_1 + a_2 + \ldots + a_n), o \in f(a, "d.s_1.s_2.....s_m"))\};$$
$$\mu = \{(x, y) \mid \exists u \exists a (u \in V(G_{in}), u[N] = "d", a \in u[O], |f(a, "d.s_1.s_2.....s_m")| \geq (a_1 + a_2 + \ldots + a_n), x \in \&f(a, "d.s_1.s_2.....s_m"), y = \&a))\}$$

This means that, for each of new produced vertices (i.e., $d_1, d_2, \ldots, d_n$), its content objects O are all obtained from the path expansion for the objects' collection in the vertex "d" ($u[N] =$ "d", $a \in u[O]$, $o \in f(a, "d.s_1.s_2. \ldots s_m")$) and its element R points out the particular immediate ancestor objects for these content objects.

Example 1: Based on the instantiation shown in Fig. 1, we illustrate the scan and structure expansion operators. Given the scan operation ϵ["m←MOVIE"](Ø), its output is shown as Fig. 2a. For the structure expansion operation, η["clip(1) IN m.video.clips"](G1), its output is shown as Fig. 2b.

Data filter operators: Monadic selection and binary selection: After obtaining content information from the multimedia databases, we now need to filter all the content

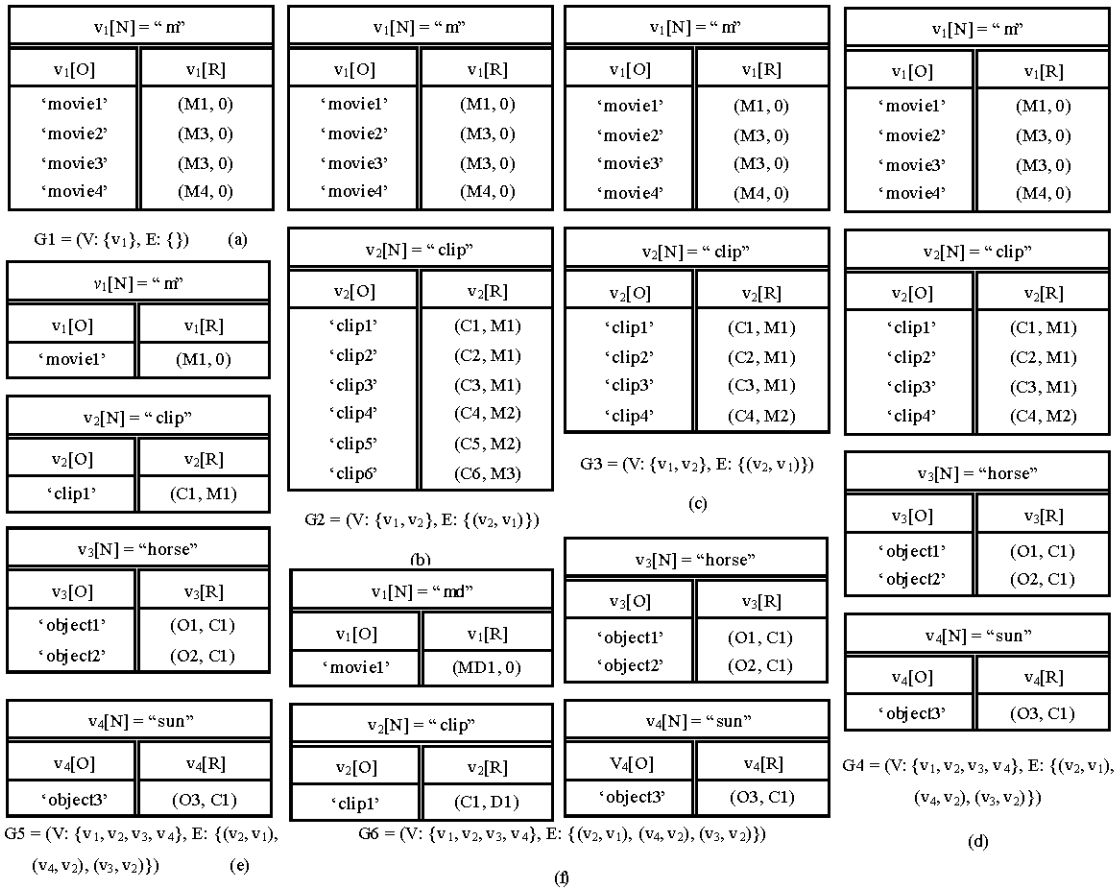

Fig. 2: Implementation results of UMQA operations

objects not satisfying the given conditional restrictions. For that, the operators of normal selection (σ^{NL}), feature selection (σ^{FE}) and spatio-temporal selection (σ^{SP}) are designed to cut the tips of branches for the input query generation graphs, i.e., to filter the content objects in the vertex variables but not satisfying selection conditional restrictions.

Remark 1: Let V(F) be a set of variables contained in the conditional expression F, each generally declared in a structure conditional item or in the FROM clause. For example, V("clip1 BEFORE[T] clip2") = {"clip1", "clip2"}.

Some different with SQL, a basic selection conditional item in the WHERE clause of a UMQL query may contain one or two variables, i.e., a conditional item correlative with two variables may not be a join one. For example, the basic conditional item "horse BEFORE[Y] sun" in example 1 contains two variables (horse and sun), but it isn't a join item. UMQA implements two different operators for the two categories of selection operations, i.e., monadic selection and binary selection. We first discuss the monadic selection.

Definition 5: If ($\emptyset \subset D \subseteq 2^0$, $\forall d(d \in D \rightarrow \forall a \forall b(a, b \in d \rightarrow t(a) = t(b))$), where, t(a) is to evaluate the data type of an object a and E is a set of basic conditional items for normal selection, feature selection or spatio-temporal selection and $\forall e(e \in E \rightarrow |V(e)| = 1)$, then the mapping g: $D \times E \rightarrow D$, is a monadic selection function and for $d \in D$ and $e \in E$, g(d, e) represents a collection comprising of all the content objects belonging to the objects' collection d and satisfying the basic conditional item e. Obviously, $g(d, e) \subseteq d$.

Definition 6: Given a basic selection conditional item F, if V(F) = {"d"}, then the selection operator on the conditional item $\sigma[F](G_{in})$ (σ may be a normal selection (σ^{NL}), feature selection (σ^{FE}) or spatio-temporal selection (σ^{SP})), outputs a new query generation graph G_O as follows:

$$V(G_O) = V(G_{in}) \cup \{v:(N:"d", O:\{o \mid \exists u\,(u \in V(G_{in}), u[N] = "d", o \in g(u[O], "F"))\},$$
$$R:\{(x,y) \mid x \in \&O, \exists u\,(u \in V(G_{in}), u[N] = "d", (x,y) \in u[R])\})\} - \{u \mid u \in V(G_{in}), u[N] = "d"\};$$
$$E(G_O) = E(G_{in})$$

This means, the selection operator on only one variable (namely monadic selection) outputs a new query generation graph in which the vertex named by "d" has been altered to remove those content objects not satisfying the given selection conditional item (u[N] = "d", o∈g(u[O], "F")) and delete the ancestor-child relationships correlative with the removed objects at the same time. In our specifications, such alterations are described as removing the original graph vertex and then inserting a new vertex with the same name. Besides, it should be noted that in a query generation graph the name of each vertex is exclusive, so it's also as the vertex identifier. Therefore in G_O, the name of the new inserted vertex "d" is accordant with that of the original removed vertex (both "d"), so that the edge's set of the graph doesn't need to be altered (both $E(G_{in})$).

If a selection conditional item contains two variables, then its operation is relatively more complicated. Let u_1 and u_2 be the two variables contained by a binary selection conditional item, the processing of its operation can be illuminated briefly as follows. First all the content objects in u_1 or u_2 are classified into several groups, making in every group all the content objects of the same ancestor. Next for each group g_1 of objects in u_1 and each group g_2 in u_2, if both have the same ancestor, then generate a pair of object's groups (g_1, g_2). Last we apply the binary selection predicate contained by the input conditional item to each pair of object's groups to remove all those objects not satisfying the binary condition.

Definition 7: If $(\emptyset \subset D \subseteq 2^0 \times 2^0, \forall(a, b)((a, b) \in D \rightarrow \forall p\,\forall q\,(p \in, a, q \in b \rightarrow t(p) = t(q))))$, E is a set of basic conditional items for a feature selection or spatio-temporal selection and $\forall e(e \in E \rightarrow |V(e)| = 2)$ and N is a set of natural number, then the mapping h: $D \times (N \times N) \times E \rightarrow D$, is a binary selection function and for $(d = (a, b)) \in D$, $e \in E$ (P is the binary selection predicate contained in e) and $(n, m) \in N \times N$, have $h(d, (n, m), e) = (a', b')$, where, $a' = \{t \mid \exists \alpha\ \exists \beta\ (t \in \alpha, \alpha \subseteq a, |\alpha| = n, \beta \subseteq b, |\beta|) = m \rightarrow \forall p \forall q\ (p \in \alpha, q \in \beta \rightarrow P(p, q)))\}$; $b' = \{t \mid \exists \alpha \exists \beta\ (\alpha \subseteq a, \beta \subseteq b, t \in \beta, |\alpha| = n, |\beta|) = m \rightarrow \forall p \forall q\ (p \in \alpha, q \in \beta \rightarrow P(p, q)))\}$. We respectively note a' with h (d, (n, m), e) (1) and b' with h(d, (n, m), e)(2).

Definition 8: Given a selection conditional item F, if V(F) = {"d_1", "d_2"}, then the selection operator on the conditional item $\sigma[F](G_{in})$ (σ may be a feature selection (σ^{FE}) or spatio-temporal selection (σ^{SP})), based on the input operand G_{in}, outputs a new query generation graph G_O as follows:

$$V(G_O) = V(G_{in}) \cup \{v_1:(N:"d_1", O:\mu, R:\{(x,y) \mid x \in \&O, \exists u\,(u \in V(G_{in}), u[N] = "d_1", (x,y) \in u[R])\}),$$
$$v_2:(N:"d_2", O:\rho, R:\{(x,y) \mid x \in \&O, \exists u\,(u \in V(G_{in}), u[N] = "d_2", (x,y) \in u[R])\})\}$$
$$-\{u \mid u \in V(G_{in}), u[N] = "d_1" \text{ or } "d_2"\};$$
$$E(G_O) = E(G_{in})$$

This means, the selection operator on two variables (namely binary selection) outputs a new query graph, where the original vertices with the name "d_1" or "d_2" are replaced by two new vertices, in which the objects collections O are refreshed by ρ and μ, respectively and the corresponding ancestor-child relationships in R are removed. And ρ and μ are defined as follows, where, N(v) is to evaluate the correlative number for the variable v (Definition 2).

$$\mu = \{o \mid \exists u \exists v\,(u, v \in V(G_{in}), u[N] = "d_1", v[N] = "d_2",$$
$$\exists a \exists b\,(a \subseteq u[O], b \subseteq v[O], \forall p \forall q(p \in a, q \in b,$$
$$\exists t\,((\&p, t) \in a[R], (\&q, t) \in b[R])), o \in h((a,b), (N("d_1"), N("d_2")), "F")(1)))\};$$
$$\rho = \{o \mid \exists u \exists v\,(u, v \in V(G_{in}), u[N] = "d_1", v[N] = "d_2",$$
$$\exists a \exists b\,(a \subseteq u[O], b \subseteq v[O], \forall p \forall q(p \in a, q \in b,$$
$$\exists t\,((\&p, t) \in a[R], (\&q, t) \in b[R])), o \in h((a,b), (N("d_1"), N("d_2")), "F")(2)))\}.$$

This means to remove every objects' group a with the same ancestor, stored in the variable "d_1", but there doesn't exist a group b in the variable "d_2" to satisfy the binary predicate P(a, b) and similar is for the variable "d_2".

Example 2: Given the monadic feature selection operation σ^{FE}["frame (clip) > 55"](G2), if in Fig. 2b only frame('clip1') >55, frame ('clip2') >55, frame ('clip3') >55 and frame('clip4') >55, then the operation outputs a query generation graph shown as Fig. 2c. Given the binary spatio-temporal selection operation σ^{SP} ["horse BEFORE[Y] sun"](η["horse(2),sun(1) IN clip. objects"](G3)), if only 'object1' BEFORE[Y] 'object3' and 'object2' BEFORE[Y] 'object3', then its output is shown as Fig. 2d.

Structure selection and join: For a query statement Q, we assume it contains a structure item F = $d_1(a_1)$, $d_2(a_2)$, …, $d_n(a_n)$ IN $d.s_1.s_2.\ldots.s_m$, some monadic items F_1, F_2, …, F_k ($k \geq 1$) and some binary selection items F_1', F_2', …, F_t' ($t \geq 1$). Note the variables declared in F as U, i.e., U = {d_1, d_2, …, d_n}. If $V(F_i) \subseteq U$ (for i = 1, 2, …, k) and $V(F_i') \subseteq U$ (for i = 1, 2, …, t), then in any semantically correct UMQA plan that is equivalent with the query Q, there must be the placement order, $\eta[F] \rightarrow (\sigma[F_1], \sigma[F_2], \ldots, \sigma[F_k]) \rightarrow (\sigma'[F_1'], \sigma'[F_2'], \ldots, \sigma'[F_t']) \rightarrow \sigma^{SE}[F]$, where σ may be η, σ^{SE}, σ^{NL}, σ^{FE}

or σ^{SP} and σ' may σ^{FE} or σ^{SP}, in particular decided by its input operating expression.

This placement is determined by the query processing logic and the operator meanings. For the query Q, to judge whether any object in the variable d is target one, an expansion operator ($\eta[F]$) should be first to expand the object for obtaining its child objects; then some selection operators ($\sigma[F_1],\ldots,\sigma[F_k],\sigma'[F_1'],\ldots,\sigma'[F_t']$) from these child objects select some objects' groups, each satisfying the same conditional restriction and last we only need to check whether the size of each objects' group is accordant with that declared in F. The study is completed by a structure selection operator.

Definition 9: Given a structure conditional item $d_1(a_1)$, $d_2(a_2)$, …, $d_n(a_n)$ IN $d.s_1.s_2.\ldots.s_m$ ($n\geq 1$, $m\geq 1$), the structure selection operator on the conditional item σ^{SE}["$d_1(a_1)$, $d_2(a_2)$, …, $d_n(a_n)$ IN $d.s_1.s_2.\ldots.s_m$"](G_{in}), based on the input operand G_{in}, outputs a new generation graph G_O as follows:

$$V(G_O)=V(G_{in})\cup\{v:(N:"d",O:\{o\mid\exists u(u\in V(G_{in}),u[N]="d",o\in u[O],\forall i(1\leq i\leq n\rightarrow\mid\{a\mid\exists v(v\in V(G_{in}),v[N]="d_i",\exists(x,y)(x=\&a,y=\&o,(x,y)\in v[R]))\}\mid\geq a_i))\},R:\{(x,y)\mid x\in\&O,\exists u(u\in V(G_{in}),u[N]="d",(x,y)\in u[R]))\})\}$$
$$-\{u\mid u\in V(G_{in}),u[N]="d"\};\ E(G_O)=E(G_{in})$$

This means to check that, for every object in the variable "d", in its expansion along the path $d.s_1.s_2.\ldots.s_m$ whether there are more than $(a_1+a_2+\ldots+a_n)$ child objects, besides, where a_1 objects belong to the child variable "d_1", a_2 objects of the rest belong to the child variable "d_2" and so on.

UMQA join is similar to that of the relational algebraic system (except with different operands). Moreover, for a UMQL query, the join variables should be declared in the FROM clause and the join attributes should be of normal data type, for UMQA, which represents that join operation should occur between two variables without any ancestor.

Definition 10: Given a join conditional item F, if V(F) = {"d_1", "d_2"} and P is the join predicate, then the join operator on the conditional item $\Pi[F](G_{ina}, G_{inb})$ (assume that $d_1\in G_{ina}$ and $d_2\in G_{inb}$), based on its input G_{ina} and G_{inb}, outputs a new query generation graph G_O as follows:

$$V(G_O)=V(G_{ina})\cup V(G_{inb})\cup(\{v:(N:"d_1\circ d_2",O:\mu,R:\{(x,y)\mid x\in\&O,y=0\})\}$$
$$-\{u\mid(u\in V(G_{ina}),u[N]="d_1")\text{ or }(u\in V(G_{inb}),u[N]="d_2")\}\cup(\beta-\alpha);$$
$$E(G_O)=E(G_{ina})\cup E(G_{inb})\cup\{(u,v)\mid v\in V(G_O),v[N]="d_1\circ d_2",\exists r(((u,r)\in E(G_{ina}),r[N]="d_1")\text{ or }((u,r)\in E(G_{inb}),r[N]="d_2"))\}$$
$$-\{(u,v)\mid((u,v)\in E(G_{ina}),v[N]="d_1")\text{ or }((u,v)\in E(G_{inb}),v[N]="d_2")\}$$

This means to combine the variables "d_1" and "d_2" into the same variable "$d_1\circ d_2$" and alter the variables originally connecting to "d_1" or "d_2" to connect to the new variable "$d_1\circ d_2$". And:

$$\mu=\{\widehat{ab}\mid\exists u\exists v(u\in V(G_{ina}),u[N]="d_1",a\in u[O],v\in V(G_{inb}),v[N]="d_2",b\in v[O],P(a,b)=\text{true})\};$$
$$\alpha=\{u\mid\exists v(((u,v)\in E(G_{ina}),v[N]="d_1")\text{ or }((u,v)\in E(G_{inb}),v[N]="d_2"))\};\ \beta=\{u\mid\exists v(v\in\alpha,v[N]=u[N],v[O]=u[O],u[R]=\{(x,y)\mid\exists\widehat{ab}(\widehat{ab}\in\mu,a\in v[O],x=\&a,y=\&\widehat{ab})\})\}$$

This means that, for the new variable "$d_1\circ d_2$", its composite content objects are selected by the join predicate P from the Cartesian product of two objects' collections, respectively stored in "d_1" and "d_2" and for all the vertices originally connecting to "d_1" or "d_2", each of their objects is refreshed by newly pointing to the corresponding composite object in "$d_1\circ d_2$" (α is the original vertices and β is the new vertices in which R are newly defined.).

Example 3: Given the following structure selection operations, σ^{SE}["clip(1) IN m.video.clips"](σ^{SE}["horse(2), sun(1) IN clip.objects"](G4)), the output is shown as Fig. 2e. Then given the join operation, Π["m.director = d.id"] (G5, ε ["d←PERSON"]), its implementing result is shown as Fig. 2f.

From definition 10, we know that, in a join output graph, the new variable name is the combination of two join variables, so for a selection operator, it's possible that in its operand there isn't any variable whose name is accordant with that contained in its operating expression, especially when it located on the outer level of a UMQA plan. For example, if the join in example 1 is implemented before, then in the operands of its subsequent selections, there is a variable named by "m∘d" instead of "m". Hence, the "=" operation on two variable names referred in the definitions of this section should be redefined.

Remark 2: If a and b are the names of two variables and assume that a = "d_0" and b = "$d_1\circ d_2\circ\ldots\circ d_n$" ($n\geq 1$), where d_0, d_1, d_2, … and d_n are all characters strings, then "a = b" or "b = a" is true, if and only if, $\exists i\ (1\leq i\leq n, d = d_0)$.

TRANSLATION FROM UMQL TO UMQA

For UMQA, its basic language elements are a powerful set of algebraic operators, employed to construct a variety of UMQA plans, such that UMQA owns equivalent capability with UMQL on multimedia query specification; however, as an operator-based algebraic language, UMQA is more suitable for internal

query processing, query optimization and query implementation. Therefore, a UMQL query should be converted into its equivalent UMQA plan. In this section, we give the construction on UMQA plan for a UMQL query and illustrate it by some examples.

UMQA plan construction: A UMQA plan consists of a series of operators, which is one solution scheme of the execution for a UMQL query and organized in a binary tree form internally, so it is also called a UMQA query plan tree. In a UMQA plan, each node represents a UMQA operation and each ordered edge represents an implementation partial order between nodes. For example, in a UMQA plan, if there is an edge (N, N') that starts from the node N to N', then we must implement the UMQA operation N before N'. Now we briefly summarize the construction process of a UMQA plan. For simplicity, we below note the node that produce a variable u as Node(u). we know that Node(u) must be an operator of scan or structure expansion.

UMQA query plan trees are all constructed bottom-up. The algorithm to construct the query plan tree for a UMQL query uses the following steps. The first step is the construction of the nodes of scan operators that are located in the bottom of the query plan tree. All variables declared in the FROM clause of a UMQL query statement are top-level ones, each without any ancestor variable. Thus for every variable declaration item E in the FROM clause, we generate a scan operator that uses E as its scan operating conditional expression and then add these scan operators into the query plan tree as its leaf nodes.

Next, split the conjunctional conditional expression of the WHERE clause into several basic conditional items. For each conditional item E, if it contains two variables (noted as u_1 and u_2) and both are declared in the FROM clause, then it is a join conditional item and thus a join operator using E as its input join conditional expression is generated and added to the query plan tree. At the same time, two edges respectively from the node Node(u_1) and Node(u_2) to the new generated join node are generated and added into the query plan tree and all other edges that originally start from Node(u_1) or Node(u_2) are altered to start from the join node. Node(u_1) and Node(u_2) should be both scan operators.

In a semantically correct UMQA query plan, from the definitions presented earlier, we know that a structure expansion and its structure selection must appear together. The third step is to construct structure expansions and their corresponding structure selections. For each structure conditional item E of the WHERE clause, we generate two nodes, a structure expansion N_1 and its corresponding structure selection N_2, both using E as their input operating conditions. Assume the expanded variable in E be u, i.e., $u \in V(E)$. Then two edges that, respectively start from Node(u) to the structure expansion node N_1 and from N_1 to its structure selection N_2 are generated and added to the query plan tree. At the same time, all the edges that originally start from Node(u) are altered to start from N_2. Please note that the variable expanded by a structure expansion should be generated by a scan or another structure expansion. Moreover, from the operator placement order given earlier, we know in the third step, all the structure conditional items of the WHERE clause should be handled as the determinate order, i.e., for two arbitrary structure conditional items E_1 and E_2 of the WHERE clause, if the operating variable of E_1 is declared in E_2, then the conditional item E_1 should be handled before E_2.

The fourth step is to construct the nodes of monadic selection operators and the fifth step is to construct the nodes of binary selection operators. For each conditional item E that contains only one variable u in the WHERE clause, we generate a monadic selection node N, whose particular operation may be a normal, feature, or spatio-temporal selection. Then we generate an edge from Node(u) to the node N and add the edge into the query plan tree. And all the edges originally from Node(u) are altered to start from N. UMQL prescribes that the operating variable of a monadic selection conditional item is allowable to be declared in the WHERE clause or a structure conditional item, but the variables of a binary selection conditional item must be both declared in the same structure conditional item. Hence, the construction of binary selections is some different with that of monadic selections. It first generates a node N of binary selection for each selection conditional item E operating on two variables in the WHERE clause, then generates an edge that start from the node N to the structure selection node N' of the query plan tree in which the two variables contained by E are both declared together and last alters all the edges that originally ended up with the structure selection node N' to end with the new generated node N. The sixth step is to construct the node of output operator.

We only give the construction of a UMQA plan for a basic external UMQL query statement, so the next step is to construct child plan trees for every internal query statement, which however can be handled using the similar steps with the front. Therefore, for each internal query statement, we generate a query plan tree and add it into the result query plan. All these child query plans can be treated as some selection operators on the variables declared in external conditional items, so the process of adding them into the result tree is handled similarly to the fourth step.

UMQA plan instances: In the plan construction, for compactness and comprehensibility, we overleap the processing for other clauses (e.g., ORDER BY clause, GROUP BY clause etc.) and assume that the input query statement owns correct syntax and semantics and all its language words have been extracted accurately. The work actually is completed by the grammar analyzer, whose design and implementation discussed detailedly by Cao *et al*. (2008) and Huang (2008), which mainly consists of a syntax analyzer and a semantic analyzer, respectively to check the syntax and semantic restrictions for any UMQL query and to extract all key words. To illustrate the translation, we below give three UMQA plans, based on the data instance in Fig. 1.

Example 4: Find movies whose posters contain three salient content objects, two 'horse' and one 'sun'.

SELECT m.name, m.video, m.poster FROM MOVIE m
WHERE horse(2), sun(1) IN m.poster.objects AND is(horse, 'horse') AND is(sun, 'sun')

Its equivalent query plan is presented as follws:

π ["m.name, m.video, m.poster"](_σ^{SE} ["horse(2), sun(1) IN m.poster.objects"](_
σ^{FE} ["is(horse,'horse')"](σ^{FE} ["is(sun,'sun')"](
η ["horse(2), sun(1) IN m.poster.objects"]
(_$\in$ ["m←MOVIE"](Ø))))))

In example 4, the plan contains five operators. We below explain its processing logic. First a collection of content objects 'MOVIE' in multimedia database is retrieved and bound into a variable 'm' (i.e., scan). Next based on a path "m.poster.objects", all the objects in 'm' are expanded to obtain all their child objects and bound into two variables, 'horse' and 'sun', respectively (i.e., structure expansion). Thirdly, we filter the content objects in 'horse' and 'sun' that don' t satisfy the feature restrictions (i.e., feature selection). Fourthly, every content object in 'm' is removed if it not containing enough residual child objects in 'horse' and 'sun' (i.e., structure selection). Finally, the residual content objects in 'm' are all target ones. In the plan, the absence of normal selection and spatio-temporal selection is allowable and in fact the absence for any category of selections is also allowable.

Example 5: Find movies directed by 'Ang Lee' and satisfying the following conditions

- There exist one video clip that contains more than fifty five video frames in each of these movies
- There are three content objects, two 'horse' and one 'sun', contained by each of these video clips

SELECT m.name, m.video, d.name FROM MOVIE m, PERSON d
WHERE d.name = 'Ang Lee' AND d.role = 'director' AND d.id = m.director AND clip(1) IN m.video.clips AND horse(2), sun(1) IN clip.objects AND frame(clip) > 55 AND is(horse, 'horse') AND is(sun, 'sun')

Its equivalent query plan is presented as follws:

π ["m.name, m.video, d.name"] (σ^{SE} ["clip(1) IN m.video.clips"] (σ^{SE} ["horse(2), sun(1) IN clip.objects"](
σ^{FE} ["is(horse, 'horse')"] (σ^{FE} ["is(sun, 'sun')"] (σ^{FE} ["frame(clip) > 55"](
σ^{NL} ["d.name = 'Ang Lee'"](η ["horse(2), sun(1) IN clip.objects"](η ["clip(1) IN m.video.clips"](_
Π["d.id = m.director"]($\in$ ["m←MOVIE"](Ø), $\in$ ["d←PERSON"](Ø)))))))))))

In example 5, the join operator combines the two variables 'm' and 'd' into the same variable and name it 'm◦d', so its subsequent structure expansion η["clip(1) IN m.video.clips"] would act on the variable 'm◦d' instead of 'm'. Moreover, in this example, there are two structure conditional items, E_1 = "clip(1) IN m.video.clips" and E_2 = "horse(2), sun(1) IN clip.objects". According as the plan construction algorithm, both should be handled as a determinate order, i.e., E_1 should be handled before E_2, which insures the semantic correctness of the final query plan.

Example 6: Find movies directed by Ang Lee and satisfying the condition as follows:

- There are two video clips in each of the movies, where the first clip contains more than fifty five video frames
- There exist two salient content objects in the first video clip, two 'horse', whose color features are similar to that of the image, 'horse1.bmp' and shape features are similar to that of the image, 'horse2.bmp'
- There exists a salient content object in the second video clip, 'sun', besides, whose shape feature is similar to that of the given image, 'sun.bmp'
- Last, the second video clip appears before the first one

SELECT m.name, m.video, d.name FROM MOVIE m, PERSON d
WHERE d.name = 'Ang Lee' AND d.id = m.director AND d.role = 'director '

AND clip1(1), clip2(1) IN m.video.clips AND horse(2) IN clip1.objects AND sun(1) IN clip2.objects
AND frame(clip1) > 55 AND is(horse, 'horse') AND is(sun, 'sun') AND color(horse, 'horse1.bmp') > 0.75 AND shape(horse, 'horse2.bmp') > 0.75 AND shape(sun, 'sun.bmp') > 0.75 AND clip2 BEFORE[T] clip1

Its equivalent query plan is presented as follws:

π ["m.name, m.video, d.name"](σ^{SE} ["clip1(1), clip2(1) IN m.video.clips"](
σ^{SE} ["horse(2) IN clip1.objects"](σ^{SE} ["sun(1) IN clip2.objects"](
σ^{SP} ["clip1 BEFORE[T] clip2"](
σ^{FE} ["is(horse, 'horse')"](σ^{FE} ["is(sun, 'sun')"](
σ^{FE} ["color(horse, 'horse1.bmp') > 0.75"] (σ^{FE} ["shape(horse, 'horse2.bmp') > 0.75"](
σ^{FE} ["shape(sun, 'sun.bmp') > 0.75"] (σ^{FE} ["frame(clip1) > 55"](_
σ^{NL} ["d.name = 'Ang Lee'"](
η ["horse(2) IN clip1.objects"](_η ["sun(1) IN clip2.objects"] (η ["clip1(1), clip2(1) IN m.video.clips"](
Π["d.id = m.director"] (∈ ["m←MOVIE"](Ø), ∈ ["d←PERSON"](Ø)))))))))))))))))

While mapping UMQL into UMQA, a structure conditional item is translated to a structure expansion and a structure selection, which is decided by the semantic feature of a structure conditional item. For instance, from the structure conditional item, "horse(2) IN clip1.objects", we know that, for any object in the variable "clip1", it is target one, if and only if, its collection attribute "clip1.objects" contains two child objects that satisfy those given restrictions (such as "is(horse, 'horse')"). So in a UMQA plan, structure expansion and structure selection always appear together and generally between both there are some other normal, feature and spatio-temporal selection operators.

UMQA TRANSLATION FORMULAS

For UMQA, we earlier only define a powerful set of algebraic operators and however, a complete algebraic system still should contain a set of operation rules among these algebraic operators, i.e., equivalent algebraic translation formulas. UMQA operators are formally similar to the relational operators, but factually, both own different meanings, so a majority of traditional equivalent translation formulas for the relational algebraic system are obviously no longer applicable for UMQA. Moreover, the UMQA plans immediately generated by the plan construction algorithm are generally not of the optimal performance, therefore, which all need to be optimized and however, such algebraic optimization must abided by equivalent algebraic translation formulas, especially when algebraic rewriting; otherwise, it would lead to semantically incorrect query plans. So for these, in this section, we present a powerful set of equivalent translation formulas specialized for UMQA.

Two plans E_1 and E_2, equivalent with each other, denotes that both have the same logical meanings, namely whose implementations on the same operand would output the same result. We note it as $E_1 \equiv E_2$. We give a complete set of equivalent translation formulas specialized for UMQA, based on the logical meanings of UMQA operators, mainly including the conjunction laws and exchange laws among algebraic operators.

Remark 3: For a basic structure conditional item $d_1(a_1)$, $d_2(a_2)$, …, $d_n(a_n)$ IN $d.s_1.s_2.\ldots.s_k$ ($n \geq 1$, $k \geq 1$), d_1, d_2, … and d_n are called former variables and d is called a latter variable. Let $V_L(F)$ and $V_F(F)$ respectively be a set of former variables and a set of latter variables, contained by a structure conditional expression F.

Conjunction laws of projection or expansion: The conjunction formula among projections means that, to eliminate attributes from an objects' collection in series is equivalent to eliminate all these attributes once, except for the projected ones:

$$\pi[A_1](E) \equiv \pi[A_1](\pi[A_2](\ldots(\pi[A_n](E))\ldots)), \quad \text{if } \forall i(1 \leq i \leq n-1 \rightarrow A_i \subseteq A_{i+1}) \tag{1}$$

where, E is a plan and A_i is a set of attributes for i = 1, 2, …, n ($n \geq 1$).

When a composite structure expansion is translated into some basic expansion operations, it's necessary to arrange these basic expansions as a determinate order, for being equivalent to the original operation, so the conjunction formula among structure expansions is given as follows:

$$\eta[E_1, E_2, \ldots, E_n](E) \equiv \eta[E_1](\eta[E_2](\ldots(\eta[E_n](E))\ldots)), \quad \text{if } \forall a(1 \leq a < n \rightarrow \forall b(1 \leq b < a \rightarrow V_L(E_a) \not\subset V_F(E_b))) \tag{2}$$

where, E is a plan and E_i is a basic structure conditional item for i = 1, 2, …, n ($n \geq 1$).

For two structure conditional items E_1 and E_2, if $V_L(E_2) \subseteq V_F(E_1)$, $\eta[E_1]$ is called the foregoing expansion of $\eta[E_2]$. Equation 2 requires that, if a query plan contains some structure expansions, each expansion must be

located after all its foregoing expansions. For instance, for $\eta[E_1, E_2](E)$ in Example 5 (E_1 = "clip(1) IN m.video.clips" and E_2 = "horse(2), sun(1) IN clip.objects"), its equivalent non conjunctional operation is $\eta[E_2](\eta[E_1](E))$; otherwise, if an expansion is located before its any foregoing expansion, the expansion operation wouldn't obtain its operating variable in its operand, which is obviously not allowable from definition 4.

Conjunction and exchange laws of selection: For some adjoining basic operators of normal selection, feature selection or spatio-temporal selection, it is allowable to combine them into the same conjunctional operation. The conjunction formulas for these selections are given as follows:

$$\sigma[F_1 \wedge F_2 \wedge \ldots \wedge F_n](E) \equiv \sigma[F_1](\sigma[F_2](\ldots(\sigma[F_n](E))\ldots)), \quad \text{if } \sigma = (\sigma^{NL}, \sigma^{FE}, \text{or } \sigma^{SP}) \tag{3}$$

$$\sigma^{SE}[F_1 \wedge F_2 \wedge \ldots \wedge F_n](E) \equiv \sigma^{SE}[F_1](\sigma^{SE}[F_2](\ldots(\sigma^{SE}[F_n](E))\ldots)), \quad \text{if } \forall a \forall b(1 \le a < b \le n \rightarrow V_L(F_a) \not\subset V_F(F_b)) \tag{4}$$

where, E is a plan and F_i is a basic selection conditional item for i = 1, 2, …, n (n≥1).

For two structure selection conditional items F_1 and F_2, if $V_L(F_2) \subseteq V_F(F_1)$, $\sigma^{SE}[F_2]$ is called the foregoing structure selection of $\sigma^{SE}[F_1]$, which is just contrary to structure expansion. Equation 4 requires that, if a query plan contains some structure selections, then any of them should be located after all its foregoing structure selections. This placement order is determined by definition 9, in which, for a variable, its structure selection depends on some operations on its child variables, so its structure selection must be performed after all these child variables complete themselves structure selections (if there are some structure selections on these child variables).

For two adjoining selection operators, their exchange laws are classified into several instances:

- If both belong to normal selection, feature selection or spatio-temporal selection, then both are immediate exchangeable
- If both are structure selections, then both are conditional exchangeable
- If one is structure selection and the other is normal selection, feature selection or spatio-temporal selection, then both are conditional exchangeable. Hence, the exchange formulas among selection operations are given as follows:

$$\sigma[F_1](\sigma''[F_2](E)) \equiv \sigma''[F_2](\sigma[F_1](E)), \quad \text{if } \sigma', \sigma'' = (\sigma^{NL}, \sigma^{SP}, \text{or } \sigma^{FE}) \tag{5}$$

$$\sigma'[F_1](\sigma''[F_2](E)) \equiv \sigma''[F_2](\sigma'[F_1](E)), \quad \text{if } \sigma' = \sigma^{SE}; \sigma'' = (\sigma^{NL}, \sigma^{FE}, \text{or } \sigma^{SP}); V_F(F_1) \cap V(F_2) = \varnothing \tag{6}$$

$$\sigma^{SE}[F_1](\sigma^{SE}[F_2](E)) \equiv \sigma^{SE}[F_2](\sigma^{SE}[F_1](E)), \quad \text{if } V_L(F_2) = V_L(F_1) \tag{7}$$

$$\sigma^{SE}[F_1](\sigma^{SE}[F_2](E)) \equiv \sigma^{SE}[F_1 \wedge F_2](\sigma^{SE}[F_1](E)), \quad \text{if } V_L(F_2) \subseteq V_F(F_1) \tag{8}$$

where, E is a plan and, F_1 and F_2 are both selection conditional expressions.

Equation 4 requires that some basic structure selections translated from a composite one should be arranged as a determinate order; however, Eq. 8 indicates that, if changing the arrangement order, the operating expression of each operator should be readjusted. For instance, for $\sigma^{SE}[F_1 \wedge F_2](E)$ in example 5 (F_1 = "clip(1) IN m.video.clips" and, F_2 = "horse(2), sun(1) IN clip.objects"), we know that, from Eq. 4, its equivalent non conjunction operation is $\sigma^{SE}[F_1](\sigma^{SE}[F_2](E))$, but from Eq. 8, its another equivalent operation is $\sigma^{SE}[F_1 \wedge F_2](\sigma^{SE}[F_1](E))$. Essentially, such restriction is also determined by definition 9.

Exchange lawa of selection and other operators: For a structure expansion and a normal, feature or spatio-temporal selection, if there aren't a variable that is not only contained by the selection operating expression but also declared in the expansion operating expression, then both are exchangeable:

$$\eta[F'](\sigma[F](E)) \equiv \sigma[F](\eta[F'](E)), \quad \text{if } \sigma = (\sigma^{NL}, \sigma^{FE}, \text{or } \sigma^{SP}); V_F(F') \cap V(F) = \varnothing \tag{9}$$

where, E is a plan and F' and F are both operating expressions.

The exchange formula between structure expansion and structure selection means that, for a structure expansion and a structure selection, if the intersection of both former variables and the intersection of both latter variables are empty, then they are exchangeable:

$$\eta[A](\sigma^{SE}[F](E)) \equiv \sigma^{SE}[F](\eta[A](E)), \quad \text{if } V_F(A) \cap V_F(F) = \varnothing; V_L(A) \cap V_L(F) = \varnothing \tag{10}$$

where, E is a plan and, F' and F are both structure conditional expressions. The operator placement order referred earlier indicates that, for a structure expansion and a selection, if there is a variable contained by both operating expressions, their arrangement order is restricted and described as Eq. 9 and 10.

For a normal, feature or spatio-temporal selection (or a structure expansion) and a projection, if adjoining with each other, then both are exchangeable, described as:

$$\pi[A_1](\zeta[F](E)) \equiv \pi[A_1](\zeta[F](\pi[A_1, A_2](E))), \quad \text{if } \zeta = (\eta, \sigma^{NL}, \sigma^{FE}, \sigma^{SE}, \text{or } \sigma^{SP}) \tag{11}$$

where, E is a query plan; F is an operating conditional expression; A_1 and A_2 are two attributes' sets and each attribute of A_2 is contained in F but not belongs to A_1.

Exchange laws of join and other operators: If E_1 and E_2 are both query plans and F is a join operating expression, the exchange formula among joins is given as follows:

$$\Pi[F](E_1, E_2) \equiv \Pi[F](E_2, E_1) \tag{12}$$

If E_1, E_2 and E_3 are all query plans and F_1 and F_2 are both join operating expressions, then the associative formula among joins is given as follows:

$$\Pi[F_2](\Pi[F_1](E_1, E_2), E_3) \equiv \Pi[F_1](E_1, \Pi[F_2](E_2, E_3)) \tag{13}$$

It is effortless to demonstrate Eq. 12 and 13 from definition 10. The exchange formula and associative formula among joins indicate that, for some joins adjoining with each other, their placement order is optional.

The exchange law on selection and join can be classified into three instances. i.e., for a normal, feature, structure or spatio-temporal selection σ,

(1) If the attributes involved in its operating expression F are all contained in a plan E_1, then we have:

$$\sigma[F](\Pi[F'](E_1, E_2)) \equiv \Pi[F'](\sigma[F](E_1), E_2) \tag{14}$$

where, F' is a join conditional expression and E_2 is a plan.

(2) if $F = F_1 \wedge F_2$, the attributes involved in F_1 all belong to E_1 and the attributes involved in F_2 all belong to E_2, then from Eq. 3 to 8 and Eq. 14, we have:

$$\sigma[F_1 \wedge F_2](\Pi[F'](E_1, E_2)) \equiv \Pi[F'](\sigma[F_1](E_1), \sigma[F_2](E_2)) \tag{15}$$

(3) if the attributes involved in F_1 all belong to E_1 and the attributes in involved F_2 all belong to E_1 or E_2, we have:

$$\sigma[F_1 \wedge F_2](\Pi[F'](E_1, E_2)) \equiv \sigma[F_2](\Pi[F'](\sigma[F_1](E_1), E_2)) \tag{16}$$

Analogously, for adjoining structure expansion and join,

(1) If the latter variables contained in the structure operating expression are only correlative with one join operand, then both are exchangeable:

$$\eta[F](\Pi[F'](E_1, E_2)) \equiv \Pi[F'](\eta[F](E_1), E_2) \tag{17}$$

where, F is a structure conditional expression, F' is a join conditional expression and, E_1 and E_2 are both plans

(2) or else if F = "F_1, F_2" and the latter variables involved in F_1 are only correlative with E_1 and the latter variables involved in F_2 are only correlative with E_2, then we have:

$$\eta[F_1, F_2](\Pi[F'](E_1, E_2)) \equiv \Pi[F'](\eta[F_1](E_1), \eta[F_2](E_2)) \tag{18}$$

The exchange formula for projection and join is given as follows:

$$\pi[A_1, A_2](\Pi[F](E_1, E_2)) \equiv \Pi[F](\pi[A_1](E_1), \pi[A_2](E_2)) \tag{19}$$

where, F is a join conditional expression, E_1 and E_2 are two plans and, A_1 and A_2 are two sets of attributes, but the attributes of A_1 only concern E_1 and the attributes of A_2 only concern E_2.

In Eq. 18 and 19, we only present some basic equivalent translation formulas specialized for UMQA; however, based on them, some more complex translation formulas can be generated. When query optimization, the algebraic translation on top of these equivalent formulas can insure the semantic correctness of query plans and consequently establish the foundation for internal query processing optimization.

SYSTEM AND IMPLEMENTATION

Prototype system: We have implemented a prototype system for multimedia information retrieval and coded UMQL and UMQA into the system, respectively as its external query language and internal processing algebra, which runs on Windows XP or Windows NT and was developed using C++ and Win32 API for its system server and Visual Basic for its some clients. The system uses the client server methodology into its architecture, i.e., it consists of a system server, many clients and socket application program interfaces used to connect the server with the clients (Fig. 3, where those functional server components that haven't been completed are shaded with points). we briefly summarize the main features of the prototype system.

The system server is the most essential component, whose function is to process UMQL queries from many clients. More specifically, it would check the grammar

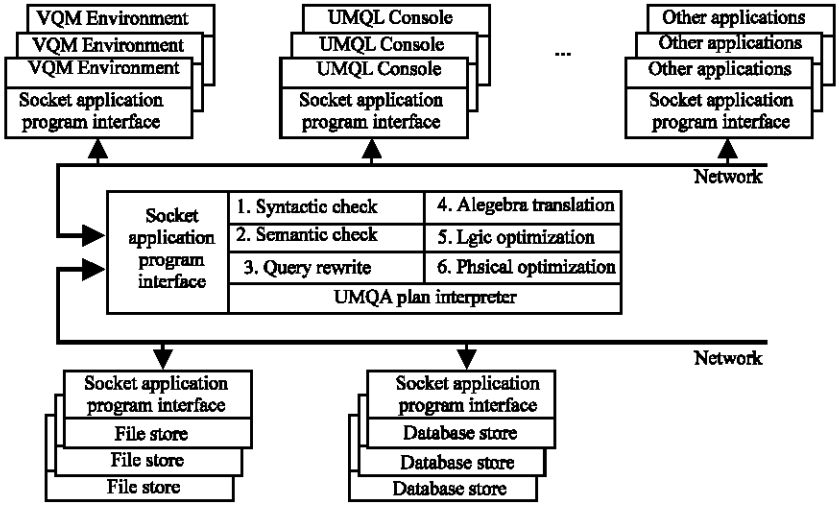

Fig. 3: Framework of UMQL-based prototype information system

correctness of a UMQL query; translate the UMQL query if owning correct syntax and semantics to its equivalent UMQA plan; optimize the UMQA plan to produce its optimal query plan; interpret and implement all operators of the query plan bottom-up and last return the query results to clients. All these functions are completed by a series of child components such as syntax analyzer, semantic analyzer, algebraic translator, algebraic optimizer and plan interpreter. Moreover, the server is a multi-threaded application, i.e., for each request from clients, it would create a service thread.

In the system, there may be many clients that all connect to the UMQL server using the socket application program interface. The common functions of these clients are to supply a friendly interface for users and display the multimedia presentations received from the server. All clients that we presently have implemented include a textual environment (i.e., UMQL console) and a VMQ visual environment, where the former implementation is relatively effortless and the latter is based on an icon-based visual query language called VMQ (Wu *et al*., 2008), of the equivalent ability with UMQL on multimedia query specification. Moreover, there exist a category of special clients in the system, to store multimedia data and their content information, so all called storage clients, including File Store based on OS file system and Database Store based on database system. When loading the system server, all the multimedia contents stored in the storage clients would be retrieved and loaded once into the server buffer but not including multimedia original data, which are returned to users only when playing after completing retrieval.

Operator implementation: In the prototype system, the plan interpreter component is employed to interpret and implement the plans generated by the translator and optimizer for obtaining target content information from multimedia databases efficiently, so whose essentials are the efficient implementations for all operators in a query plan. In most traditional database systems, such implementations are generally completed by some index techniques. However in UMQA, for some of its operators that are accordant or similar with those of the traditional relational algebraic system, such as join and normal selection, their implementations can be completed by the traditional index methods, but for the other operators, such as structure expansion and binary feature or spatio-temporal selection, from their specifications given earlier, we know that, the traditional implementation methods are no longer applicable. So we need to restudy their efficient implementations. We below briefly summarize the implementation for these operators.

For a structure expansion, the operation on expanding its input path expression include traversals of many additional intermediate collections of content objects in multimedia databases, so it is time consuming and may lead to inefficient implementations. Based on the observation that, all content objects don't hold too many generations children (generally not more than 10 generations), we propose a code scheme for each object in the multimedia databases and then create index on top of these object codes. In this code scheme, we assign unique codes to every content object in multimedia database, such that, for every content object, the prefix of its code equals to the code of any of its ancestor content

objects, i.e., given a content object u, if its code is C(u), then the code for any immediate child object v of u is C(v) = C(u)• n (n≥1 and n is the serial number of the content object v in all the immediate child objects of u). So, this is a prefix code scheme also called Dewey decimal classification code (Chien *et al*., 2002) that has been widely applied. Using these codes, we can judge whether two arbitrarily given content objects contain an ancestor-child relationship effortlessly. We then use the traditional B^+ tree index method to create cluster index for every objects' collection of the same constructed data type, on top of these prefix codes. After completing such code index creation, then in the implementation for expanding any content object a based on a path expression "$d.s_1.s_2. \ldots. s_m$", we first obtain the code range (C(a), C(a) + 1) for the child content objects of a and then combining the B^+ index tree, we can from the objects' collection corresponding to "$d.s_1.s_2. \ldots. s_m$" obtain all the content objects whose prefix codes are located in (C(a), C(a) + 1), where C(a) represents the prefix code of a given content object a. Such implementation using the code index scheme can obtain the entire child content objects for an expanded objects' collection by immediately accessing the target collection in multimedia database and consequently can avoid to read those additional intermediate objects' collections. This is obviously more efficient.

For a binary feature or spatio-temporal selection, from definition 8, we know that, the most essential problem on its implementation is to evaluate the binary selection function constructed based on the selection operating conditional item efficiently and however, from definition 7, we know that, such binary selection function evaluation is generally expensive, because of it containing the binary feature or spatio-temporal selection predicate P, whose implementations are complicated and quite expensive. So for implementing a binary selection operation efficiently, we should reduce the number of computing its corresponding binary selection predicate, as well as the search space of evaluating its binary selection function, as much as possible. From Definition 3.7 on binary selection function, if we construct a unordered graph G as follows:

- Use a' and b', the output of a binary selection function, as the set of vertices
- Based on the connectivity of each vertex in a' with each in b', construct the set of edges, then we know that, such graph G must be a bipartite graph and it is comprising of a series of child (nm) complete bipartite graphs

Therefore, the problem of evaluating the binary selection function is equivalent to search all child complete bipartite graphs. We can present some useful properties on bipartite graph and using them, improve the performance of searching child bipartite graphs and consequently reduce the search space for evaluating the binary selection function. Moreover, to reduce the number of evaluating the expensive binary predicate P, we use a buffer technique, i.e., we define the adjacency matrix for the bipartite graph G, which only apply the binary selection predicate P once for each object in a with each in b (a and b are both the input of binary function), consequently avoiding to compute P repeatedly. In conclusion, in the implementation for binary feature or spatio-temporal selection operations, we use the bipartite graph theory to illuminate the essentials on evaluating the binary selection function and then use some correlative properties to reduce its evaluation cost.

Presently, we have coded these implementation approaches into our prototype information system and have acquired acceptable application effects. Due to space constraints, more particular implementations and corresponding test reports will be given in another study.

CONCLUSION AND FUTURE WORK

In this study, we have presented an operator-based algebra called UMQA, for internally representing and processing UMQL queries. UMQA is formally similar to the traditional relational algebra, both of a powerful set of similar algebra operations. To deal with UMQL querying on multimedia contents of structure, feature and spatio-temporal relationship, UMQA is introduced with some new operators, i.e., structure selection (σ^{SE}), structure expansion (η), feature selection (σ^{FE}) and spatio-temporal selection (σ^{SP}), which make UMQA of equivalent ability with UMQL on multimedia query specification, i.e., for any UMQL query, UMQA can construct its equivalent query plan. We then present an approach to transform UMQL queries into their equivalent UMQA plans. A complete algebraic system still should contain a set of operation rules among operators, i.e., equivalent algebraic translation formulas, which is also very important for query optimization when algebraic rewriting. So in this study, we have also presented a powerful set of equivalent algebraic translation formulas, specialized for UMQA, mostly concentrating on the conjunction laws and exchange laws among operators, consequently, establishing the foundation for subsequent algebraic optimization.

However, the UMQA plans generated by the construction algorithm given in this study can't be insured of the best performance, namely which need to be optimized. Based on the equivalent translation formulas given in this study, a plan can be translated into large numbers of equivalent plans, which is the exponential of

the number of operators in the plan. Hence, it is time consuming to search the optimal plan by exhausting all possible instances and consequently it is necessary to give some heuristic optimization strategies to reduce the search space. However, it is more unfortunate that some traditional heuristic optimization strategies are no longer suitable for optimizing UMQA plans. For instance, in the relational algebraic system, selection pushdown is the most essential heuristic strategy, in which under the assumption of the low expenses of selections, selections have typically been handled by "pushdown" in order to reduce subsequent join execution costs. However in UMQA, its selections generally contain some expensive predicates (such as feature function and spatio-temporal predicates), so the assumption of traditional optimization methods does not come into existence and its optimization strategies is also no longer applicable. Therefore, the optimization for UMQA plans needs to be restudied. All of these are next work and will be discussed in another study.

ACKNOWLEDGMENT

This research is partially supported by the National 863 High-Tech Research and Development Plan of China under Grant Nos. 2006AA01Z430.

REFERENCES

Allen, J.F., 1983. Maintaining knowledge about temporal intervals. Commun. ACM, 26: 832-843.

Aygun, R.S. and A. Yazici, 2004. Modeling and management of fuzzy information in multimedia database applications. Multimedia Tools Appli., 24: 29-56.

Balkir, N., H.G. Ozsoyoglu and Z.M. Ozsoyoglu, 2002. A graphical query language: VISUAL and its query processing. IEEE Trans. Knowledge Data Eng., 14: 955-978.

Cao, Z.S., Z.D. Wu and Y.Z. Wang, 2007. UMQL: A unified multimedia query language. Proceedings of the IEEE International Conference on Signal Image Technology and Internet Based Systems, Dec. 2007, Shanghai, China, pp: 101-107.

Cao, Z.S., Z.D. Wu and Y.Z. Wang, 2008. A grammar analysis model for the unified multimedia query language. J. Elect. Sci. Technol. China, 6: 87-93.

Chien, S.Y., Z. Vagena and D. Zhang, 2002. Structural joins: A primitive for efficient XML query pattern matching. Proceedings of the IEEE International Conference on Data Engineering, 2002, San Jose, USA, pp: 141-152.

Christel, M.G. and A.G. Hauptmann, 2005. The use and utility of high-level semantics features in video retrieval. Lect. Notes Comput. Sci., 3568: 134-144.

Huang, D.W., 2008. Study on Query Analysis for a Multimedia Query Language UMQL. Huazhong University of Science and Technology, Wuhan, China.

Lee, T., L. Sheng, T. Bozkaya, N.H. Balkir, Z.M. Ozsoyoglu and G. Ozsoyoglu, 1999a. Querying multimedia presentations based on content. IEEE Trans. Knowledge Data Eng., 11: 361-385.

Lee, T., Z.M. Ozsoyoglu and G. Ozsoyoglu, 1999b. A graph query language and its query processing. Proceedings of the IEEE International Conference on Data Engineering, Mar. 1999, Sydney, Australia, pp: 1-1.

Lee, T., L. Sheng and N.H. Balkir, 2000. Query processing techniques for multimedia presentations. Multimedia Tools Appli., 11: 63-99.

Li, J. and Z.M. Ozsoyoglu, 1996. Processing OODB queries by o-algebra. Proceedings of the ACM International Conference on Information and Knowledge Management, Nov. 1996, Rockville, Maryland, USA, pp: 1-1.

Liu, Y., D.S. Zhang, G.J. Guo and W.Y. Ma, 2007. A survey of content-based image retrieval with high-level semantics. Pattern Recogn., 40: 262-282.

Meng, X.F., Y. Wang and X.F. Wang, 2006. Research on XML query optimization. Chinese J. Software, 17: 2069-2087.

Shaw, G.M. and S.B. Zdonik, 1990. A query algebra for object-oriented databases. Proceedings of the International Conference on Data Engineering, Feb. 5-9, IEEE Computer Society Washington, DC, USA., pp: 154-162.

Sophie, C. and D. Claude, 1992. A general framework for the optimization of object-oriented queries. Proceedings of the ACM SIGMOD International Conference on Management of Data, 1992, ACM New York, USA., pp: 383-392.

Tian, Z.P., H.R. Dang and A.Y. Zhou, 1999. Multimedia object query language and its query processing. Chinese J. Software, 10: 694-701.

Wang, Y., L.Z. Zhou and C.X. Xing, 2007. Video semantic models and their evaluation criteria. Chinese J. Comput., 30: 337-340.

Wu, Z.D., Z.S. Cao and Y.Z. Wang, 2008. Design and implementation of a visual multimedia query language. J. Huazhong Univ. Sci. Technol., 36: 45-56.

Zuo, Q. and Z.S. Cao, 2008. G3M: A generalized multimedia data model based on MPEG-7. Proceedings of the International Conference on Multimedia and Ubiquitous Engineering, Busan, Apr. 24-26, IEEE Computer Society Washington, DC, USA., pp: 155-159.