Journal of Southeast University (English Edition) Vol. 25, No. 3, pp. 320 – 325 Sept. 2009 ISSN 1003—7985

Processing and optimization of UMQL-based multimedia queries

Wu Zongda Cao Zhongsheng Wang Yuanzhen Li Guiling

(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: Through the mapping from UMQL (unified multimedia query language) conditional expressions to UMQA (unified multimedia query algebra) query operations, a translation algorithm from a UMQL query to a UMQA query plan is put forward, which can generate an equivalent UMQA internal query plan for any UMQL query. Then, to improve the execution costs of UMQA query plans effectively, equivalent UMQA translation formulae and general optimization strategies are studied, and an optimization algorithm for UMQA internal query plans is presented. This algorithm uses equivalent UMQA translation formulae to optimize query plans, and makes the optimized query plans accord with the optimization strategies as much as possible. Finally, the logic implementation methods of UMQA plans, i. e., logic implementation methods of UMQA operators, are discussed to obtain useful target data from a multimedia database. All of these algorithms are implemented in a UMQL prototype system. Application results show that these query processing techniques are feasible and applicable.

Key words: multimedia database; multimedia query language; query optimization; unified multimedia query language

As a powerful tool for specifying users' query requirements, a multimedia query language is one of the most essential components in a multimedia database management system. In Refs. [1 – 3], we designed a general-purpose multimedia query language called UMQL. UMQL allows users to query multimedia data based on content information such as structure, feature and spatio-temporal relationship, and thus it is expressive and competent for multimedia information retrieval. For internal representation, we subsequently designed an operator-based algebraic language called UMQA in Ref. [4]. It has equivalent ability with UMQL on multimedia query specification. Although there exist many query processing proposals[5–8], they are not applicable because they are designed for specific query languages or particular applications. In this paper, we propose some processing and optimization techniques for UMQL-based multimedia queries. Given a UMQL query having correct syntax and semantics, the translation into an equivalent UMQA plan, the optimization of the UMQA plan to minimize its execution costs, and the interpretation and implementation of the plan to obtain target multimedia information from a multimedia database are discussed.

Received 2009-01-05.

Biographies: Wu Zongda (1983—), male, graduate; Cao Zhongsheng (corresponding author), male, doctor, professor, caozhongsheng@163.com.

Foundation item: The National High Technology Research and Development Program of China (863 Program) (No. 2006AA01Z430).

Citation: Wu Zongda, Cao Zhongsheng, Wang Yuanzhen, et al. Processing and optimization of UMQL-based multimedia queries[J]. Journal of Southeast University (English Edition), 2009, 25(3): 320 – 325.

1 Translation from UMQL to UMQA

UMQL is based on a semi-structured data organization model. It includes basic notions such as constructed data type, collection data type, object, and child object. A constructed data type, whose instance is an object, is a composite structure of predefined data types (e. g., FLOAT, INTEGER, etc.), collection data types and other constructed data types. The instance of a collection data type is a composite value of one or more elements of the same data type. Each basic item of an object is an attribute, whose value is called an attribute value of the object; it is also called a child object of the object when the data type of the basic item is a collection data type or a constructed data type. UMQL uses the same SELECT-FROM-WHERE statement as SQL, but it extends the WHERE clause with structure expression, feature expression and spatio-temporal expression to accommodate complex multimedia query requirements. To show the syntax features of UMQL, we consider the following example of query movies based on video contents. The query retrieves the movies directed by Ang Lee, each video of which contains one video clip. The video clip has not only more than 55 video frames, but also three salient objects including two "horse" and one "sun". The "horse" has more than 75% color feature similarity with the image "horse. bmp", and the "sun" is located above the "horse". This query requirement can be described by UMQL as follows:

Example 1

```
SELECT m. name FROM MOVIE m, PERSON d
WHERE clip(1) IN m. video. clips
AND horse(2), sun(1) IN clip. objects//structure expression
AND d. id = m. dir AND d. name = 'Ang Lee'
AND d. role = 'director' AND frame(clip) > 55
AND is(horse, "horse") AND is(sun, "sun")
AND color(horse, 'horse. bmp') > 0. 75
AND shape(sun, 'sun. bmp') > 0. 75//feature expression
AND horse BEFORE[Y] sun//spatio-temporal expression
```

As can be seen, in the structure expression we first describe the structure of the multimedia data by declaring some new variables('m', 'clip', 'horse', 'sun') and the corresponding relationships among these variables. Secondly, based on some feature functions('frame', 'is', 'color'), a feature expression is used to define the semantic notions and bottom-level features for salient content objects represented by variables. Finally, a spatio-temporal expression is presented to define the temporal relationship between the variables 'horse' and 'sun'.

UMQA[4], including a set of operators and translation for-

mulae, is an internal algebra designed for representing and processing UMQL query internally. It includes the following operators: join(Π), normal selection(σ^{NL}), structure selection(σ^{SE}), feature selection(σ^{FE}), spatio-temporal selection (σ^{SP}), structure expansion (η) and scan(ε). The former five operators are the implementations of join expression, normal expression, structure expression, feature expression and spatio-temporal expression. And the sixth one is used to expand objects to obtain their child objects based on a structure expression. Hence, UMQA has the ability equivalent to UMQL in multimedia query specification. A UMQA plan, which internally is organized in the form of a binary tree, is a solution for the execution of a UMQL query.

Definition 1 A UMQA plan $T_P = (V, E)$ consists of $V(T_P)$, a set of nodes, and $E(T_P)$, a set of ordered pairs of distinct elements of $V(T_P)$. Each node in $V(T_P)$ represents an operation, and the operator belongs to $\{\sigma^{NL}, \sigma^{SE}, \sigma^{FE}, \sigma^{SP}, \eta, \Pi, \varepsilon\}$. Each edge (N, M) of $E(T_P)$ represents an execution order between the two operations N and M; i. e., the operation N should be implemented before M.

Algorithm 1 Translation from UMQL to UMQA

Input: a UMQL query Q of correct syntax and semantics.

Output: an equivalent UMQA query plan $T_P = (V, E)$.

1) A UMQA query plan is constructed; i. e., $V(T_P) \leftarrow \varnothing$, $E(T_P) \leftarrow \varnothing$.

2) For each variable declaration item E_0 in the FROM clause, a scan operation is constructed by using E_0 as the operating expression and put into $V(T_P)$.

3) The conjunctional conditional expression of the WHERE clause is split into five parts F_1, F_2, F_3, F_4 and F_5, which are comprised of basic join, structure, normal, feature and spatio-temporal conditional items, respectively.

4) For each join conditional item E_0 in F_1, a join operation N_0 is constructed by using E_0 as its operating expression and put into $V(T_P)$. In $V(T_P)$, if the two scan nodes which produce the join variables contained in E_0 are N and N', two edges (N, N_0) and (M, N_0) are generated and put into $E(T_P)$ while other edges that originally start from N or M are changed to start from N_0.

5) A structure expansion N_2 is constructed by using F_2 as the operating expression and put into $V(T_P)$. Supposing that the current root node of T_P is N_1, an edge (N_1, N_2) is generated and put into $E(T_P)$.

6) A normal selection operation N_3 is constructed by using F_3 as the operating expression and put into $V(T_P)$. Then, an edge (N_2, N_3) is generated and put into $E(T_P)$.

7) A feature selection operation N_4 is constructed by using F_4 as the operating expression and put into $V(T_P)$. Then, an edge (N_3, N_4) is generated and put into $E(T_P)$.

8) A spatio-temporal selection N_5 is constructed by using F_5 as the operating expression and put into $V(T_P)$. Then, an edge (N_4, N_5) is generated and put into $E(T_P)$.

9) A structure selection N_6 is constructed by using F_2 as the operating expression and put into $V(T_P)$. Then, an edge (N_5, N_6) is generated and put into $E(T_P)$.

10) A projection N_7 is constructed based on the SELECT clause and put into $V(T_P)$. Then, an edge (N_6, N_7) is generated and put into $E(T_P)$.

11) Return the UMQA query plan $T_P = (V, E)$.

In algorithm 1, we omit the processing for other clauses (e. g., ORDER BY clause), and only give the plan construction for a basic UMQL query statement. By using algorithm 1, the query in example 1 is translated into the query plan given as Fig. 1, where N01 to N09 respectively represent the following operations: ε["m←MOVIE"], ε["d←PERSON"], Π["d. id = m. dir"], η["clip IN m. video. clips AND horse(2), sun(1) IN clip. objects"], σ^{NL}["d. name = 'Ang Lee' AND d. role = 'director'"], σ^{FE}["frame(clip) > 55 AND is(horse, 'horse') AND is(sun, 'sun') AND color(horse, 'horse. bmp') > 0. 75 AND shape(sun, 'sun. bmp') > 0. 75"], σ^{SP}["horse BEFORE[Y] sun"], σ^{SE}["clip IN m. video. clips AND horse(2), sun(1) IN clip. objects"], and π["d. name"].

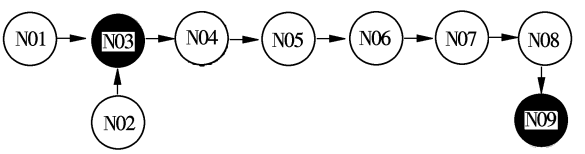

Fig. 1 Query plan produced by algorithm 1

2 Optimization for UMQA Plan

The logic optimization for a UMQA plan consists of two parts, optimization strategies and translation formulae. In the optimization of plans, we must abide by some translation formulae for maintaining the plans semantically correct. If two plans E_1 and E_2 are equivalent, then they have the same logical meanings; i. e., their implementations on the same operand will output the same result. It can be noted as $E_1 \equiv E_2$. Based on the meanings of UMQA operators, a complete set of equivalent translation formulae for UMQA is given as follows:

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

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

$$\left.\begin{array}{l} \sigma[F_1 \wedge \ldots \wedge F_n](E) \equiv \sigma[F_1](\ldots(\sigma[F_n](E))\ldots) \\ \quad \text{if } \sigma = (\sigma^{NL}, \sigma^{FE}, \text{ or } \sigma^{SP}) \\ \sigma^{SE}[F_1 \wedge \ldots \wedge F_n](E) \equiv \sigma^{SE}[F_1](\ldots(\sigma^{SE}[F_n](E))\ldots), \\ \quad \text{if } \forall a \forall b(1 \leqslant a < b \leqslant n \rightarrow V_L(F_a) \not\subset V_F(F_b)) \end{array}\right\} \tag{3}$$

$$\left.\begin{array}{l} \sigma[F_1](\sigma'[F_2](E)) \equiv \sigma'[F_2](\sigma[F_1](E)) \\ \quad \text{if } \sigma, \sigma' = (\sigma^{NL}, \sigma^{SP}, \text{ or } \sigma^{FE}) \\ \sigma^{SE}[F_1](\sigma[F_2](E)) \equiv \sigma[F_2](\sigma^{SE}[F_1](E)) \\ \quad \text{if } \sigma = (\sigma^{NL}, \sigma^{FE}, \text{ or } \sigma^{SP}),\ V_F(F_1) \cap V(F_2) = \varnothing \\ \sigma^{SE}[F_1](\sigma^{SE}[F_2](E)) \equiv \sigma^{SE}[F_2](\sigma^{SE}[F_1](E)) \equiv \\ \sigma^{SE}[F_1 \wedge F_2](\sigma^{SE}[F_1](E)) \qquad \text{if } V_L(F_2) = V_L(F_1) \end{array}\right\} \tag{4}$$

$$\left.\begin{array}{l} \eta[F'](\sigma[F](E)) \equiv \sigma[F](\eta[F'](E)) \\ \quad \text{if } \sigma = (\sigma^{NL}, \sigma^{FE}, \text{ or } \sigma^{SP}),\ V_F(F') \cap V(F) = \varnothing \\ \eta[A](\sigma^{SE}[F](E)) \equiv \sigma^{SE}[F](\eta[A](E)) \\ \quad \text{if } V_F(A) \cap V_F(F) = \varnothing,\ V_L(A) \cap V_L(F) = \varnothing \end{array}\right\} \tag{5}$$

$$\pi[A_1](\zeta[F](E)) \equiv \pi[A_1](\zeta[F](\pi[A_1, A_2](E))) \quad \text{if } \zeta = (\eta, \sigma^{NL}, \sigma^{FE}, \sigma^{SE}, \text{ or } \sigma^{SP}) \tag{6}$$

$$\Pi[F](E_1, E_2) \equiv \Pi[F](E_2, E_1) \tag{7}$$

$$\Pi[F_2](\Pi[F_1](E_1, E_2), E_3) \equiv \Pi[F_1] \cdot (E_1, \Pi[F_2](E_2, E_3)) \tag{8}$$

$$\left.\begin{array}{l}\sigma[F](\Pi[F'](E_1, E_2)) \equiv \Pi[F'](\sigma[F](E_1), E_2) \\ \sigma[F_1 \wedge F_2](\Pi[F'](E_1, E_2)) \equiv \Pi[F'](\sigma[F_1](E_1), \sigma[F_2](E_2)) \\ \sigma[F_1 \wedge F_2](\Pi[F'](E_1, E_2)) \equiv \sigma[F_2](\Pi[F'](\sigma[F_1](E_1), E_2))\end{array}\right\} \tag{9}$$

$$\left.\begin{array}{l}\eta[F](\Pi[F'](E_1, E_2)) \equiv \Pi[F'](\eta[F](E_1), E_2) \\ \eta[F_1, F_2](\Pi[F'](E_1, E_2)) \equiv \Pi[F'](\eta[F_1](E_1), \eta[F_2](E_2))\end{array}\right\} \tag{10}$$

$$\pi[A_1, A_2](\Pi[F](E_1, E_2)) \equiv \Pi[F](\pi[A_1](E_1), \pi[A_2](E_2)) \tag{11}$$

Rules 1 to 3 are conjunctional laws. Rules 4 to 11 are exchangeable laws. Based on these translation formulae and optimization strategies[9-10], the optimization algorithm is given as algorithm 2, where all the nodes are handled from bottom to top. Because the choice for join order and join method is established by the traditional optimization algorithm, algorithm 2 concentrates on how to optimize the placement for all selections in a plan. In the first step, complex conjunctional operations are expanded to generate some new operations. In the second step, these new operations are pushed down as much as possible. In the third step, the multimedia operation placement algorithm is used to move up each selection to a suitable position of a query plan based on the selectivity and unit execution costs of selection.

Algorithm 2 Optimization of a UMQA plan

Input: a plan produced by algorithm 1.

Output: a new plan after optimization.

1) Conjunctional complex operations are expanded.

a) Rule 2 is used to translate the structure expansion operation as

$$\eta[E_1, E_2, \ldots, E_n](E) \Rightarrow \eta[E_1](\eta[E_2](\ldots(\eta[E_n](E))\ldots))$$
$$\forall a(1 \leqslant a < n \rightarrow \forall b(1 \leqslant b < a \rightarrow V_L(E_a) \not\subset V_F(E_b)))$$

b) Rule 3 is used to translate the normal selection, feature selection and spatio-temporal selection operations as

$$\sigma[F_1 \wedge \ldots \wedge F_n](E) \Rightarrow \sigma[F_1](\ldots(\sigma[F_n](E))\ldots)$$

c) Rule 3 is used to translate the structure selection operation as

$$\sigma^{SE}[F_1 \wedge \ldots \wedge F_n](E) \Rightarrow \sigma^{SE}[F_1](\ldots(\sigma^{SE}[F_n](E))\ldots)$$
$$\forall a(1 \leqslant a < n \rightarrow \forall b(a < b < n \rightarrow V_L(E_a) \not\subset V_F(E_b)))$$

2) Operations are pushed down.

a) Rule 10 is used to push each node which represents a structure expansion operation down along the plan as far as possible.

b) Rules 5 and 9 are used to push each node which represents normal, feature or spatio-temporal selection down along the query plan as far as possible.

c) Rules 4, 5 and 9 are used to push each node which represents a structure selection operation down along the query plan as far as possible.

d) Rules 6 and 11 are used to push each node which represents a projection operation down along the plan as far as possible.

3) For each node except one representing the join operation in the query plan, the multimedia operation placement algorithm is used to push it up from root to leaf.

4) Rule 3 is used to combine the nodes which represent the selection operations and adjoin each other with a node representing a single selection whose condition is the conjunction of each original selection's condition.

5) Rule 1 is used to combine several projection operation nodes which adjoin with each other to produce a new single projection node. The conditional item of the new node is the conjunction of the original projection operations.

6) Return the query plan after optimization.

For the UMQA plan in Fig. 1, its optimization by algorithm 2 is shown as Fig. 2, where V01 to V16 respectively represent the following operations: ε ["m ← MOVIE"], π ["m. dir, m. video"], Π ["d. id = m. dir"], ε ["d←PERSON"], π ["d. name, d. id"], σ^{NL} ["d. name = 'Ang Lee' AND d. role = 'director'"], η ["clip IN m. video. clips"], σ^{FE} ["frame(clip) > 55"], σ^{SE} ["clip IN m. video. clips"], η ["horse(2), sun(1) IN clip. objects"], σ^{FE} ["is(horse, 'horse') AND color(horse, 'horse. bmp') > 0.75"], σ^{FE} ["is(sun, 'sun') AND shape(sun, 'sun. bmp') > 0.75"], σ^{SP} ["horse BEFORE[Y] sun"], σ^{SE} ["horse(2), sun(1) IN clip. objects"], σ^{SE} ["clip IN m. video. clips AND horse(2), sun(1) IN clip. objects"], and π ["d. name"]. The evolvement of the node N08 in Fig. 1, which represents a conjunctional structure selection operation, is taken as an example. In the first step of algorithm 2, N08 is split into two nodes V09 and V10, and a new edge (V14, V09) is output. In the second step, the second formula of rule 4 is used to push down the node V09 below the node V14, generating a new node V15. However, in the third step, all of them are pushed down below the join node V03. And the multimedia operation placement algorithm pushes up the nodes V09, V14 and V15 below the node V03 again, because the join in the node V03 has very low selectivity on the variable 'm' (as the movies directed by 'Ang Lee' are few). In other words, the earlier implementation of V03 can reduce the input size of its subsequent selection operations and the total execution costs.

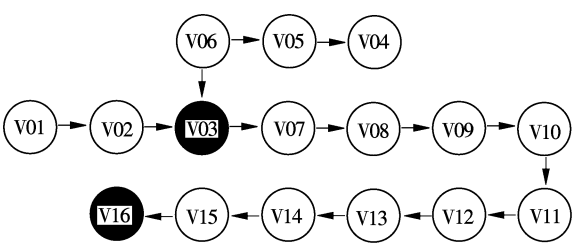

Fig. 2 Query plan optimized by algorithm 2

3 Logical Implementation for UMQA Plan

In this section, we discuss how to implement a UMQA plan so as to obtain target data from a multimedia database. In such a process, an essential data structure is a query generation graph which is used to specify the intermediate results for implementing a query plan.

Definition 2 A query generation graph $G = (V, E)$ con-

sists of $V(G)$, a nonempty set of vertices, and $E(G)$, a set of ordered pairs of distinct elements of $V(G)$.

1) Each ordered edge (u, v) of $E(G)$ represents an ancestor-child relationship between two variables denoted by the vertices u and v, and $\forall(u', v')\,\forall(u, v)((u', v') \in E \wedge (u, v) \in E \wedge u' = u \rightarrow v' = v)$.

2) Each vertex u of $V(G)$ represents a variable comprising 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 $u[R]$ is a set of ordered pairs of OIDs and used to point out the immediate ancestor object for each object in $u[O]$, satisfying

$$\left.\begin{array}{l}\forall(x, y)((x, y) \in u[R] \rightarrow x \in \&u[O] \wedge y \in \&v[O]) \\ \quad \text{if } \exists(u, v) \in E(G) \\ \forall(x, y)((x, y) \in u[R] \rightarrow x \in \&u[O] \wedge y = 0) \quad \text{otherwise}\end{array}\right\}$$

where $\&a$ represents the OID of a if a is a content object; or else it represents a collection of OIDs, each denoting an object in a, if a is a collection of content objects.

Definition 3 P and O are the domain of path expressions and the domain of objects, respectively. If $\varnothing \subset D \subseteq O \times P$, then the mapping f: $D \rightarrow 2^O$ is an expansion function. When $o \in O$ and $p \in P$, $f(o, p)$ is a collection comprising all objects and the object o can be reached along the path expression p.

Definition 4 If $(\varnothing \subset D \subseteq 2^O) \wedge \forall d(d \in D \rightarrow \forall a\,\forall b(a, b \in d \rightarrow t(a) = t(b)))$, $\forall e(e \in E \rightarrow |V(e)| = 1)$, then the mapping g: $D \times E \rightarrow D$ is a monadic selection function. Here the function t is used to evaluate the data type of an object and E is a set of basic conditional items for normal, feature or spatio-temporal selection. When $d \in D$ and $e \in E$, $g(d, e)$ represents a collection of objects satisfying the basic conditional item e in the objects' collection d. Obviously, $g(d, e) \subseteq d$.

Definition 5 If $\varnothing \subset D \subseteq 2^O \times 2^O \wedge \forall(x, y)((x, y) \in D \rightarrow \forall a\,\forall b(a \in x \wedge b \in y \rightarrow t(a) = t(b)))$, $\forall e(e \in E \rightarrow |V(e)| = 2)$, then the mapping h: $D \times (N \times N) \times E \rightarrow D$ is a binary selection function. Here E is a set of basic conditional items for feature selection or spatio-temporal selection, and N is a set of natural numbers. When $d \in D \wedge e \in E \wedge (n, m) \in N \times N \wedge d = (x, y) \wedge e = P(x, y)$, $h(d, (n, m), e) = (x', y')$, where P is a binary selection predicate, and $\forall a(a \in x' \rightarrow \exists e_1(e_1 \in y \wedge P(e_1, a)) \wedge \exists e_2(e_2 \in y \wedge P(e_2, a)) \wedge \ldots \wedge \exists e_n(e_n \in y \wedge P(e_n, a))) \wedge \forall a(a \in y' \rightarrow \exists e_1(e_1 \in x \wedge P(e_1, a)) \wedge \exists e_2(e_2 \in x \wedge P(e_2, a)) \wedge \ldots \wedge \exists e_m(e_m \in x \wedge P(e_m, a)))$.

Definition 6 If $\varnothing \subset D \subseteq 2^O \times 2^O \wedge \forall(x, y)((x, y) \in D \rightarrow \forall a\,\forall b(a, b \in x \rightarrow t(a) = t(b)) \wedge \forall a\,\forall b(a, b \in y \rightarrow t(b)))$, $\forall e(e \in E \rightarrow |V(e)| = 2)$, then the mapping z: $D \times E \rightarrow 2^{O \times O}$ is a join function, where E is a set of basic join items. When $d \in D \wedge e \in E \wedge d = (x, y) \wedge e = P(x, y)$, $z(d, e) = \{\widehat{ab} \mid a \in x \wedge b \in y \wedge P(a, b)\}$, where P is a basic join binary predicate.

Based on the above definitions, the implementation algorithm of a UMQA plan can be put forward as follows:

Algorithm 3 Implementation of a UMQA plan

Input: a UMQA plan.

Output: a query generation graph $G(V, E)$.

1) $V(G) \leftarrow \{\}$, $E(G) \leftarrow \{\}$. A post-order traverse is made for the plan and for each vertex.

2) If a scan ε["$d \leftarrow D$"] is represented, then an object set O of the same type "D" is obtained from a multimedia database. A vertex("d", O, $\{(\&a, 0) \mid a \in O\}$) is generated and added into $V(G)$.

3) If a structure expansion operation is represented, then for each basic conditional item e = "$d_1(a_1)$, $d_2(a_2)$, ..., $d_n(a_n)$ IN $d.s_1.s_2. \ldots s_k$", the following three steps are taken.

a) n vertices are generated and named as d_1, d_2, ..., d_n, where $d_i[O] \leftarrow \varnothing$, $d_i[R] \leftarrow \varnothing$. Then they are put into $V(G)$.

b) A set of objects $d[O]$ is obtained from the vertex d.

c) For each object o of $d[O]$, call $f(o, e)$ to obtain a set of child objects, Q. If $Q \mid \geqslant (a_1 + a_2 + \ldots + a_n)$, then $d_i[O] \leftarrow d_i[O] \cup Q$ and $d_i[R] \leftarrow d_i[R] \cup \{(\&a, \&o) \mid a \in Q\}(i = 1, 2, \ldots, n)$.

d) n edges $(d_1, d), \ldots, (d_n, d)$ are generated and put into $E(G)$.

4) If a normal, feature or spatio-temporal selection is represented, then for a basic conditional item e of the operation, the following two steps are taken.

a) If $|V(e)| = 1$ and $V(e) = \{d\}$, a set of objects $d[O]$ is obtained from the vertex d. Then for each $a \in d[O] - g(d[O], e)$, call procedure update (d, a).

b) If $|V(e)| = 2$ and $V(e) = \{d_1, d_2\}$, two sets of objects $d_1[O]$ and $d_2[O]$ are obtained from the vertex d_1 and d_2, respectively. Call $h((d_1[O], d_2[O]), (N(d_1), N(d_2)), e)$ and return (O_1, O_2). Then for each $a \in d_1[O] - O_1$, call update(d_1, a), and for each $b \in d_2[O] - O_2$, call update(d_2, b).

5) If a structure selection operation is represented, then for each basic conditional item, e = "$d_1(a_1)$, $d_2(a_2)$, ..., $d_n(a_n)$ IN $d.s_1.s_2. \ldots s_k$", the following two steps are taken.

a) A set of objects $d[O]$ is obtained from the vertex d.

b) For each o of $d[O]$, some sets of objects O_1, O_2, ..., O_n are obtained from the vertices d_1, d_2, ..., d_n, where $O_i = \{a \mid \exists d(d \in d_i[R] \wedge d[x] = \&a \wedge d[y] = \&o)\}$. If $|O_i| < a_i$, call update(d, o).

6) If the projection or join operation is represented, then it is processed similarly with the relational algebra processing.

Procedure update(v, o):

If $o \in v[O]$, then (1) remove o from $v[O]$; (2) for each $r \in v[R]$, if $r[x] = \&o$, then r is removed from $v[R]$; (3) if $\exists v'(v' \in V \wedge \langle v', v \rangle \in E)$, then for each$(\&a, \&b) \in v'[R]$ and $b = o$, call update(v', a).

In a UMQA query plan, scan and structure expansion are used to initialize a query generation graph. Normal selection, structure selection, feature selection, and spatio-temporal selection are used to cut the tips of branches. Join is used to reconstruct the query generation graph. Then, the projection operation as the root is used to construct and output the query results.

Example 2 For a scan operation ε["m←MOVIE"]$(\varnothing)$ given in Fig. 3, the output is shown as Fig. 4(a). For a structure expansion η["clip(1) IN m. video. clips"](G_1), the output is shown as Fig. 4(b), where G_1 is a query generation produced by the previous scan operation.

MOVIE

oid	name	dir	video.clips	…
M1	'movie1'	P1	{C1, C2, C3}	
M2	'movie2'	P1	{C4, C5}	
M3	'movie3'	P1	{C6}	
M4	'movie4'	P1	{}	

OBJECT

oid	name	type	…
O01	'object1'	'horse'	
O02	'object2'	'horse'	
O03	'object3'	'sun'	
O04	'object4'	'horse'	
O05	'object5'	'horse'	
O06	'object6'	'sun'	
O07	'object7'	'cattle'	
O08	'object8'	'sheep'	
O09	'object9'	'horse'	
O10	'object10'	'grass'	

CLIP

oid	name	objects	…
C1	'clip1'	{O1, O2, O3}	
C2	'clip2'	{O4, O5, O6}	
C3	'clip3'	{O7}	
C4	'clip4'	{O8}	
C5	'clip5'	{O9}	
C6	'clip6'	{O10}	
C7	'clip7'	{}	

PERSON

oid	name	role	…
P1	'Ang Lee'	'director'	
P2	'John'	'player'	
P3	'Tom'	'player'	
P4	'Steven'	'adaptor'	

Fig. 3 Scheme of all objects in system

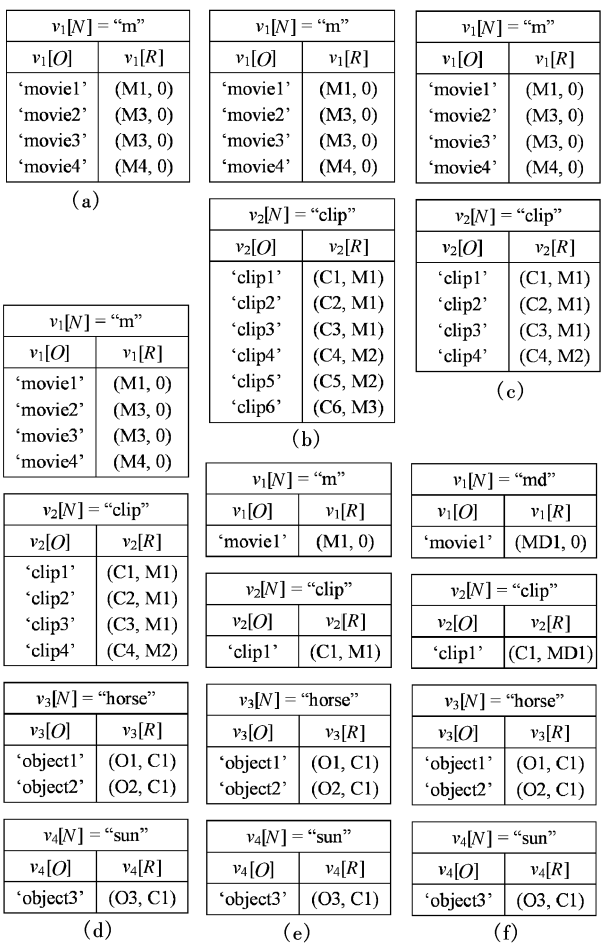

Fig. 4 Implementation results of UMQA operations. (a) $G_1 = (V: \{v_1\}, E: \{\})$; (b) $G_2 = (V: \{v_1, v_2\}, E: \{(v_2, v_1)\})$; (c) $G_3 = (V: \{v_1, v_2\}, E: \{(v_2, v_1)\})$; (d) $G_4 = (V: \{v_1, v_2, v_3, v_4\}, E: \{(v_2, v_1), (v_4, v_2), (v_3, v_2)\})$; (e) $G_5 = (V: \{v_1, v_2, v_3, v_4\}, E: \{(v_2, v_1), (v_4, v_2), (v_3, v_2)\})$; (f) $G_6 = (V: \{v_1, v_2, v_3, v_4\}, E: \{(v_2, v_1), (v_4, v_2), (v_3, v_2)\})$

Example 3 For a feature selection with only one variable σ^{FE}["frame(clip) > 55"](G_2), if only frame('clip1') > 55, frame('clip2') > 55, frame('clip3') > 55, and frame('clip4') > 55, the output is shown as Fig. 4(c). For a spatio-temporal selection with two variables σ^{SP}["horse BEFORE[Y] sun"](σ^{SP}["horse(2), sun(1) IN clip. objects"](G_3)), if there are only 'object1', 'object2', 'object3' and 'object4' satisfying 'object1' BEFORE[Y] 'object3' and 'object2' BEFORE[Y] 'object3', the output is shown as Fig. 4(d).

Example 4 For the structure selection operations σ^{SE}["clip(1) IN m. video. clips"](σ^{SE}["horse(2), sun(1) IN clip. objects"](G_4)), the output is shown as Fig. 4(e). For the join Π["m. dir = d. id"](G_5, ε["d←PERSON"]), the output is shown as Fig. 4(f).

4 Conclusion

In this paper, some UMQL-based multimedia query processing techniques are given, including the translation of a UMQL into an operator-based language UMQA, the optimization of UMQA plans, and the implementation of UMQA plans. These techniques are implemented into a UMQL prototype information system which runs on Windows XP or Windows NT and is developed by using Visual C++ and Visual Basic 6.0 for its server and clients, respectively. The prototype system is built on a client-server framework, including a system server, many clients, and socket application program interfaces used to link the server and clients, as shown in Fig. 5.

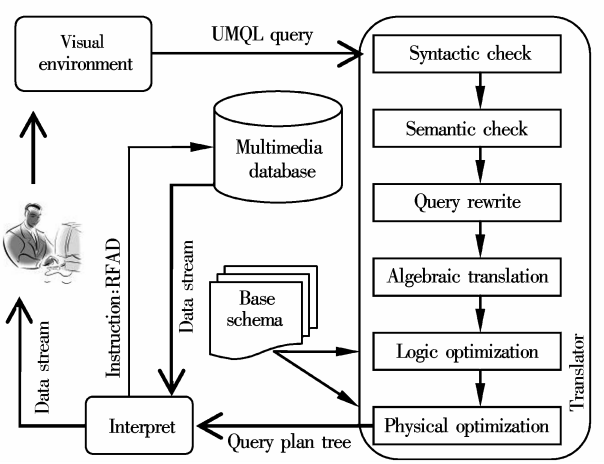

Fig. 5 Framework of UMQL prototype system

However, the crucial problems of UMQA operator implementations that exhibit an important influence on the efficiency of implementation are described in the form of some functions. The effectual implementations for these functions will be discussed in future work. And, due to space constraints, the performance test report of the algorithms is not given in this paper, which is also future work.

References

[1] Cao Zhongsheng, Wu Zongda, Wang Yuanzhen. UMQL: a unified multimedia query language[C]//*IEEE/ACM Conference on Signal Image Technology and Internet Based Systems*. Shanghai, China, 2007: 101 - 107.

[2] Wu Zongda, Cao Zhongsheng, Wang Yuanzhen. Design and implementation of visual multimedia query language[J]. *Journal of Huazhong University of Science and Technology* (*Nature Science*), 2008, **36**(7): 45 - 56. (in Chinese)

[3] Cao Zhongsheng, Wu Zongda, Wang Yuanzhen. A grammar analysis model for UMQL[J]. *Journal of Electronic Science and Technology of China*, 2008, **6**(3): 317 - 323.

[4] Wu Zongda, Cao Zhongsheng, Wang Yuanzhen. UMQA: an internal algebra for querying multimedia contents[J]. *Information Technology Journal*, 2009, **8**(4): 411 - 426.

[5] Hsu C, Knoblock C. Semantic query optimization for query plans of heterogeneous multidatabase systems [J]. *IEEE Transactions on Knowledge and Data Engineering*, 2000, **12**(6): 959 - 979.

[6] Gadia S K. Algebraic identities and query optimization in a parametric model for relational temporal databases[J]. *IEEE Transactions on Knowledge and Data Engineering*, 1998, **10**(5): 793 - 808.

[7] Grant J, Gryz J, Minker J. Logic-based query optimization for object databases[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2000, **12**(4): 529 - 548.

[8] Lee C, Chen C. Query optimization in multidatabase systems considering schema conflicts[J]. *IEEE Transactions on Knowledge and Data Engineering*, 1997, **9**(6): 941 - 956.

[9] Smith J M, Chang P Y T. Optimizing the performance of a relational algebra database interface[J]. *Communications of the ACM*, 1975, **18**(10): 568 - 579.

[10] Aho A V, Sagiv T, Jullman J D. Efficient optimization of a class of relational expressions[J]. *ACM Transactions on Database System*, 1979, **4**(4): 423 - 446.

UMQL 多媒体查询的处理和优化

吴宗大　曹忠升　王元珍　李桂玲

(华中科技大学计算机科学与技术学院, 武汉 430074)

摘要: 通过将 UMQL 查询的各类条件式映射为 UMQA 的对应代数算子, 给出从 UMQL 多媒体查询到 UMQA 查询计划的等价转换算法, 为任意 UMQL 查询生成等价的内部 UMQA 查询计划. 然后, 为了有效改善 UMQA 查询计划的执行代价, 研究了等价 UMQA 代数变换规则和一般性优化策略, 给出 UMQA 内部计划的优化算法. 该算法基于等价规则变化 UMQA 查询计划, 并使优化后的查询计划尽可能符合优化策略. 最后, 讨论了 UMQA 查询计划的逻辑执行方法, 即 UMQA 代数算子的逻辑执行方法, 以便从多媒体数据库中获取用户感兴趣的目标数据. 这些算法均在一个 UMQL 原型系统中实现, 且应用效果表明这些查询处理技术均切实可行.

关键词: 多媒体数据库; 多媒体查询语言; 查询优化; UMQL

中图分类号: TP311. 134. 3