A Grammar Analysis Model for the Unified Multimedia Query Language

Zhong-Sheng Cao, Zong-Da Wu, and Yuan-Zhen Wang

Abstract—The unified multimedia query language (UMQL) is a powerful general-purpose multimedia query language, and it is very suitable for multimedia information retrieval. The paper proposes a grammar analysis model to implement an effective grammatical processing for the language. It separates the grammar analysis of a UMQL query specification into two phases: syntactic analysis and semantic analysis, and then respectively uses Backus-Naur form (EBNF) and logical algebra to specify both restrictive grammar rules. As a result, the model can present error guiding information for a query specification which owns incorrect grammar. The model not only suits well the processing of UMQL queries, but also has a guiding significance for other projects concerning query processings of descriptive query languages.

Index Terms—Grammar analysis, query language, query processing, unified multimedia query language.

1. Introduction

With the prevalence of multimedia data, multimedia information retrieval has become one of the most important problems in the data management domains and acquired a series of achievements. A powerful multimedia query language is a useful facility to specify users' multimedia query requirements, and therefore is one of the most essential components in multimedia information query retrieval. As one of the most important traditional query facilities, the structured query language (SQL) has acquired great success, but it does not suit multimedia information retrieval because the complex spatial and temporal relationships inherent in the wide range of multimedia data types make a multimedia query language quite different from its counterpart in traditional database management systems (DBMS). In recent years, there have been many multimedia query language proposals\(^{[1]-[7]}\), which are either designed for one particular medium (e.g., images) or specialized for a particular application (e.g., a digital library), consequently not competent for multimedia information retrieval.

To implement an effective retrieval for multimedia information, we have proposed a powerful general-purpose multimedia query language called unified multimedia query language (UMQL)\(^{[8]}\), which supports queries based on the content, structure, and spatio-temporal inner information of multimedia data effectively, and has been demonstrated to be very expressive and applicable for uniform multimedia information retrieval. We have also implemented a concept prototype of UMQL based on a multimedia information base constructed on the basis of a relational database management system (RDBMS). The prototype consists of a translator and an interpreter. The translator, whose function is to generate an optimization query plan for a UMQL query, consists of several functional components, i.e., syntax analysis, semantic analysis, algebraic translation, and query optimization. The interpreter, whose function is to interpret and implement the query plan produced by the translator, obtains multimedia data from the multimedia information base, through generating read instructions in series. The prototype is presented in Fig. 1. In the prototype, the syntactic analysis and the semantic analysis, also called grammar analysis together, are some of the most critical components. The analysis processing proposals for familiar query languages presented by far are all designed for themselves applications\(^{[9]-[11]}\), and hence all of them do not suit well the processing of UMQL queries. The purpose of the present paper is to propose a grammar analysis model...
specialized for the query processing of UMQL. The results of this study are of significance to the UMQL query processing, and therefore play a critical part in our studying project on multimedia information retrieval technology.

2. Modeling Preparation

2.1 A Brief Review on UMQL

The SELECT sentence is the most essential statement in UMQL, which can support effectively queries based on content, structure, and spatio-temporal relationships of multimedia data.

Example 1. Find out movies directed by ‘Ang Lee’ and satisfying the conditions: there exist two video clips in the videos of these movies; the first clip contains two horses whose color features are similar to that of the image horse1.bmp and whose shape features are similar to that of the image horse2.bmp; the second clip contains one sun whose shape feature is similar to that of the image sun.bmp; at last, the second clip appears before the first one.

```
SELECT m.video FROM MOVIE m
WHERE clip(2) IN m.video.clips
AND horse(2) IN clip[1].objects
AND sun(1) IN clip[2].objects
AND m.author = 'Ang Lee' AND is(horse, 'horse')
AND is(sun, 'sun') AND color(horse, horse1.bmp) > 0.7
AND shape(horse, horse2.bmp) > 0.7
AND shape(sun, sun.bmp) > 0.7
```

In Example 1, SELECT, FROM, and WHERE are key words, both IN and BEFORE are predicates designed for structure conditional expressions and spatio-temporal conditional expressions respectively, and is, color, and shape are feature functions used for feature conditions.

UMQL is an extension of SQL, which introduces structure conditions, feature conditions, and spatio-temporal conditions on the basis of original conditions of SQL. The structure conditions are used to specify the inner structures of multimedia data and declare each category of multimedia child objects contained in multimedia data; the feature conditions are used to specify the bottom-level or other user-defined features for each category of multimedia child objects; the spatio-temporal conditions are used to specify the spatio-temporal relationships between these multimedia child objects. Different from its counterparts of the structured relational model of SQL and object-oriented model, the fundamental data model of UMQL is semi-structured. The data model includes several concepts such as collection data type, constructed data type, object, child object, and attribute. An instance of a collection data type is a composite value comprising of zero or more elements. A constructed data type is a composite data type comprising of several predefined data types (such as FLOAT, INT and CHAR), collection data types and other constructed data types, whose instance is called an object. Every child item in an object is an attribute, whose value is called an attribute value of the object, and also called child objects of the object if the data type of the item is a collection data type or a constructed data type.

2.2 Concept Definition

Concerning our model, there exist two main concepts: multimedia database schema and UMQL group.

Definition 1. A multimedia database schema can be described using a set, \( MDB = \{<\text{Name}, \text{Attributes}> | \text{Name} \text{ is the name of a constructed data type; \ Attributes is a collection of attributes of the constructed data type, and} \} \) \text{Type represents the data type of the attribute.}\}. So a multimedia database schema is a specification for the structure of a multimedia information base.

Definition 2. A query specification owning correct syntax in UMQL can be specified by an eightfold group, \( G = \{SE, FR, SC, ST, NC, GB, OB\} \), called a UMQL group, where:

1) \( SE = \{<\text{identifier}_{1}, \text{identifier}_{2}, \text{alias}, \text{function}> | (\text{identifier}_{1}, \text{identifier}_{2}, \text{and alias are identifiers comprising of letters and digits, and alias can be NULL.}) \times (\text{function} = \{\text{NULL, 'SUM', 'COUNT', 'MAX', 'MIN', 'AVG'}\}) \} \) which corresponds to the select clause. Each quadplex group in the SE can be assembled into a following format, \( \text{function} (\text{identifier}_{1}, \text{identifier}_{2}) \) \text{alias}. It represents a projection item of the select clause, where the \text{identifier}_{1} is an object name, the \text{identifier}_{2} is an attribute name of the object, the \text{function} represents an aggregate function, and the \text{alias} is an alias of the projection item.

2) \( FE = \{<\text{dataType}, \text{objectName}> | \text{both dataType and objectName are identifiers.} \} \) which corresponds to the from clause. For each twofold group in the \( FE \), the \text{dataType} is the name of a constructed data type, and \text{objectName} is the name of a variable bound to a set of objects, each an instance of the data type, \text{dataType}.

3) \( SC=<\text{items, expression}> \), where \( \text{items=}<\text{childObjects}, n, \text{index, path}> | (\text{childObjects is an identifier.)} \times (n = * \lor n \geq 1) \times (\text{index} \geq 0) \times (\text{path} is a list comprising of some identifiers, namely path = (\text{identifier}_{1}, \text{identifier}_{2}, \cdots, \text{identifier}_{k})), k \geq 1). \) The \( SC \) corresponds to the structure conditional expression of the \text{where} clause. Each element of the \text{items} represents a structure conditional item, \text{childObjects}(n) IN \text{identifier}_{1}. [\text{index}, \text{identifier}_{2}, \cdots, \text{identifier}_{k}]. \) The \text{expression} represents a structure conditional expression, which is organized as a binary expression tree where each leaf node represents an element of the \text{items}, and each internal node stores a logical operator used to join these conditional items.
4) \( FC = \langle items, expression \rangle \), where \( items = \{ \langle functionExpression, objects, n \rangle \mid (functionExpression \ is \ a \ string, \ of \ characters \ and \ objects \ is \ an \ identifier) \land (n \geq 0) \} \). The \( FC \) is used to specify the feature conditional expression of the \( where \) clause. Each element in the \( items \) represents a basic feature conditional item, where the \( functionExpression \) is a string of characters that represents a basic feature conditional item, and the \( objects[n] \) represents a multimedia object contained in the \( functionExpression \). The organization of the expression is similar to that of the \( SC \).

5) \( ST = \langle items, expression \rangle \), where \( items = \{ \langle objects_1, index_1, operator, direction, objects_2, index_2 \rangle \mid direction \in \{ 'X', 'Y', 'T' \} \land operator \in \{'BEFORE', 'MEETS', 'OVERLAPS', 'DURING', 'STARTS', 'FINISHES', 'EQUALS'\} \land (index_1 \geq 0) \land (index_2 \geq 0) \land (both \ objects_1 \ and \ objects_2 \ are \ identifiers) \} \). The \( ST \) corresponds to the spatio-temporal conditional expression of the \( where \) clause. Each element of the \( items \), \( objects_1[index_1] \), \( operator[direction] \), \( objects_2[index_2] \), represents a basic spatio-temporal conditional item.

6) \( NC \), \( GB \) and \( OB \) correspond to the normal conditional expression of the \( where \) clause, the \( group \) by \( clause \) and the \( order \) by \( clause \) respectively. The three categories of expressions are all compatible with those of SQL, whose processings can be implemented by the SQL processor, and accordingly the \( NC \), \( GB \) and \( OB \) are only three strings of characters same to their corresponding specifications in a UMQL query.

3. Syntactic Analysis

The grammar rules of UMQL can be separated into two parts: syntactic rules and semantic rules. The syntactic rules are a collection of restrictive rules used to specify the formal grammatical characteristics of UMQL, without considering the concrete meanings of variables in a query specification. The semantic rules are designed for specifying the logical grammatical characteristics of UMQL, which are used to check the logical validities of variables of a query specification. Accordingly, the grammar analysis model also includes two phases: syntactic analysis and semantic analysis, where the functionality of the syntactic analysis is to check the syntactic constraints of a UMQL query specification based on the syntactic rules, and then the semantic analysis will check the semantic constraints of the query specification if it owns correct syntax. The processing results of the syntactic analysis are specified using a UMQL group where each UMQL clause is separated from each other, and each variable is also drawn out from its clause, which facilitates the following semantic checking. The particular processing flow of the grammar analysis model is presented in Fig. 2.

Fig. 2. Processing flow of the UMQL grammar analysis model.

The first step of the syntactic analysis is to identify key words in a UMQL query specification, required to separate the query specification into several tokens including identifiers, key words, and special symbols. The process can be specified by a deterministic finite automation (DFA), so we don’t specify it here. The second step is to construct each clause according to \{select, from where, group by, order by\} and check their syntactic constraints based on the syntactic rules. The extended Backus-Naur from (EBNF) is an effective facility used to specify recursive context-free grammar rules, so we choose EBNF to describe the syntactic rules. The results are presented as follows.

SYNTACTIC RULE 1:

\[
<projection attributes> ::=<projection> [ ,<projection> ] ;
<projection> ::= (<function>'(<field>)*') | <field>
[ [AS] [identifier] ] | <identifier> * ;
<function> ::= SUM | COUNT | MAX | MIN | AVG ;
<field> ::= <identifier> , <identifier> ;
<identifier> ::= <letter> [ '<digit> ] ;
<digit> ::= 0 | 1 | $$ | 9 ;
<letter> ::= a | b | $$ | X | Y | Z .
\]

SYNTACTIC RULE 2:

\[
<objects declaration> ::= (<constructed type> <identifier>)
[ ,(<constructed type> <identifier>) ] ;
<constructed type> ::= <letter> [ ,<letter> , <digit> ] .
\]

SYNTACTIC RULE 3:

\[
<conditional expression> ::= <conditional term>
[ (AND | OR) <conditional term> ] ;
<conditional term> ::= <simple condition> |
[NOT] ‘<conditional expression>’ ;
<simple condition> ::= <normal condition> | <structure condition> |<feature condition> |
<spatio-temporal condition> ;
<structure condition> ::= <identifier> ‘(<digit> | * )‘ IN
<path expression> ;
<path expression> ::= <identifier> [ ,<identifier> ] ;
<feature condition> ::= <identifier function>
‘(<identifier> [ '<digit> ] , <identifier> )‘
[<comparison operator> <value>] ;
\]
SEMANTIC RULE 1:
\[ \forall \text{item} \in \text{FE} \rightarrow \exists \text{type}(\text{type} \in \text{MDB} \land \text{typeName} = \text{item}_{\text{dataType}}) \].

Explanation: 1) The first semantic rule indicates that for every feature function item in the specification of the multimedia database, there exists a type in the multimedia database schema corresponding to it.

SEMANTIC RULE 2:
\[ \forall \text{item} \in \text{SC}_{\text{items}} \land \text{item}_{\text{index}} \geq 1 \rightarrow \exists \text{term} \in \text{SC}_{\text{items}} \land \text{term}_{\text{childObjects}} = \text{item}_{\text{path}(\text{identifier})} \land \text{term}_{\text{attributes}} \geq \text{item}_{\text{index}} \land \text{type}(\text{type} \in \text{MDB} \land \text{typeName} = \text{item}_{\text{path}(\text{identifier})}) \land \exists \text{attribute}(\text{attribute} \in \text{attributes}) \land \text{attribute}_{\text{Name}} = \text{item}_{\text{path}(\text{identifier})} \land \text{IS}(\text{item}_{\text{path}(\text{identifier})}) \land \text{COLLECTION\_DATA\_TYPE}); \]

SEMANTIC RULE 3:
\[ \forall \text{item} \in \text{FC}_{\text{items}} \rightarrow \exists \text{term}(\text{term} \in \text{FC}_{\text{items}} \land \text{term}_{\text{childObjects}} = \text{item}_{\text{objects}}) \land \text{term}_{\text{attributes}} \geq \text{item}_{\text{index1}}) \land \text{COLLECTION\_DATA\_TYPE}; \]

SEMANTIC RULE 4:
\[ \forall \text{item} \in \text{ST}_{\text{items}} \rightarrow \exists \text{term}(\text{term} \in \text{ST}_{\text{items}} \land \text{term}_{\text{childObjects}} = \text{item}_{\text{objects}}) \land \text{term}_{\text{attributes}} \geq \text{item}_{\text{index1}}) \land \text{COLLECTION\_DATA\_TYPE}; \]

SEMANTIC RULE 5:
\[ \forall \text{item} \in \text{SE} \land \text{item}_{\text{identifier2}} = * \rightarrow \text{item}_{\text{function}} = \text{NULL} \land \text{function}(\text{function} = \text{NULL} \lor \text{function} = \text{COUNT}) \land \exists \text{term}(\text{term} \in \text{SE} \land \text{term}_{\text{objectName}} = \text{item}_{\text{identifier1}}); \]

Explanation: 1) The first semantic rule indicates that for every feature function item in the specification of the multimedia database, there exists a term in the multimedia database schema corresponding to it.
every constructed data type appearing in the from clause has to be an element of the multimedia database schema.  

2) In the second rule, the method, IS, is used to judge the data type of an object identifier. This semantic rule specifies the semantic constraints of the path which represents a path expression, namely the first item of a path expression has to be an object identifier declared in the from clause or in a previous structure conditional item, and other item of the path expression has to be an attribute of its parent item.  

3) The third semantic rule requires that every operating object in a basic feature conditional item should be declared in a previous structure conditional expression, where the method, CHECK_FE, is used to check whether a feature conditional characters’ string is of validity.  

4) The fourth semantic rule requires that every multimedia object contained in a basic spatio-temporal conditional item should be declared in a previous structure conditional expression, and the object contains spatio-temporal feature, where the method, CHECK_SP, is used to check whether the spatio-temporal relationships are inherent in the data type of a multimedia object.  

5) The fifth semantic rule indicates that each operating object of the projection list has to be declared in the from clause.  

6) Just as the syntactic analysis, the semantic analysis aiming at the order by clause, the group by clause and normal conditional expressions are also implemented by the SQL processor.  

Finally, the semantic analysis will print corresponding semantic error information if the query specification fails to pass the semantic checking. If a UMQL query specification owns correct syntax and semantics, its UMQL group will be sent to the next component of the language processor, the algebraic translation, which will construct the algebraic specification of the query based on the UMQL group.

5. Grammar Analysis for Nesting Queries

The previous discussions aim only at a query without nested specification (i.e., a query without child queries), so in this section we will discuss the grammar analysis for nesting queries.

From previous two sections, we know that the UMQL grammar analysis model only check the grammar constraints of the select clause, the from clause, and the structure conditional expression, feature conditional expression, and spatio-temporal conditional expression of the where clause, and the grammar analysis for the normal conditional expression, the group by clause, and the order by clause are implemented by the SQL processor. The not recursive definitions of the structure conditional expression, feature conditional expression, and spatio-temporal conditional expression make it impossible that in the three categories of expressions there exist child queries, and the abandonment of the join operator also makes it impossible to generate child queries in the from clause. So the grammar analysis for a nesting query is only required firstly to submit a query specification where the new introducing UMQL ingredients in the nesting query (e.g., the structure conditional expression etc.) have been removed to the SQL processor, then to separate the nesting query into several query blocks without nested specification if the SQL processor return successful information, and finally to check the syntactic and semantic constraints for these query blocks according previous methods. In this process, the first step implements a grammar analysis for the language ingredients compatible with SQL. The particular processing flow is presented in Fig. 3, which also applies to no nesting queries.

6. Conclusions

We have proposed a grammar analysis model for the UMQL query processing. In the model, the syntactic rules specified using EBNF and the semantic rules specified using the logical algebra are respectively presented, which makes the model able to present error guiding information for a not correct query specification. Combining the SQL processor, the model resolves successfully the grammatical processing for other language ingredients compatible with those of SQL.
The good effect of the model application indicates that the model could successfully implement a judgment of the grammar rules of UMQL. So, the methods reported here contribute to the grammatical processing of UMQL, and are the foundation of the algebraic translation and query optimization. They also play an important role in multimedia information retrieval.

References


Zhong-Sheng Cao was born in Hubei, China, in 1965. He received the B.S. degree from the Fudan University in 1989, M. S. degree and Ph.D. degree both from the Huazhong University of Science and Technology in 1992 and 1995 respectively. He is now an associate professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests are primarily in the area of multimedia database.

Zong-Da Wu was born in Zhejiang, China, in 1983. He received the B.S. degree from the College of Computer Science and Engineering, Wenzhou Normal College, Zhejiang, in 2005. He is now pursuing his Ph.D. degree with the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests are primarily in the area of multimedia query language.

Yuan-Zhen Wang was born in Hubei, China, in 1945. She is now a professor and doctor’s supervisor with the School of Computer Science and Technology, Huazhong University of Science and Technology. Her research interests are primarily in the area of database theory and technology.