VisualMOQL: A Visual Query Language for Image Databases

Vincent Oria, Bing Xu and M. Tamer Özsu
Department of Computing Science, University of Alberta
Edmonton, Alberta, Canada T6G 2H1
{*oria, bing, ozsu*}*@cs.ualberta.ca*

Abstract

Since most multimedia database systems are built on top of object or object-relational database systems, they inherit the underlying query facilities. The approach we present in this paper is in two steps. The first step is to design a multimedia query language that will be used as an internal language. The second step is to define an equivalent visual query language and a translator to translate a visual query into a query in the internal query language.

Keywords

Image database system, multimedia database system, visual query language

1 INTRODUCTION

A common solution to satisfy the diversity of multimedia data users is to provide visual techniques to retrieve multimedia data. The query is depicted by visual representations of domains of interest. This technique of expressing queries is known as *visual language*, *iconic language* or *graphical language* [ACS90]. In general, the expressive power of visual query languages are low since they are directed at naive users; they are often not based on a textual query language.

One way to extend the capabilities of visual languages is to base them on powerful multimedia query languages, which themselves may be extensions of object or object-relational query languages. This provides a visual query language that enables easy querying of multimedia databases while benefiting from the query facilities provided by the database management system (DBMS). We have defined a multimedia query language MOQL (Multimedia Object Query Language) [LÖSO97] that extends OQL (Object Query Language) [CBB+97]. In this paper, we present VisualMOQL a visual query language for the image component of MOQL defined for the DISIMA (Distributed and Interoperable Image Management System) project [OÖL+97]. The DISIMA prototype is implemented on top of the ObjectStore [LLOW91]

DBMS. The remainder of this paper is organized as follows: Section 2 introduces the DISIMA model used as a base in the VisualMOQL implementation, Section 3 gives an overview of the MOQL language, Section 4 presents VisualMOQL, and Section 5 states the conclusion.

2 THE DISIMA MODEL

The DISIMA model, is composed of two main blocks: the image block and the salient object block. We define a *block* as a group of semantically related entities.

2.1 The image block

The image block is made up of two layers: the *image* layer and the *image representation* layer. We distinguish an image from its representations to maintain an independence between them, referred to as *representation independence*. At the *image* layer, the user defines an image type classification similar to hierarchies in object type systems. This layer allows the user to define functional relationships between images. These images can be classified according to specific criteria.

2.2 The salient object block

DISIMA views the content of an image as a set of *salient objects* (i.e., interesting entities in the image) with certain spatial relationships to each other. The *salient object* block is designed to handle salient object organization. For a given application, salient objects are identified and defined. The definition of salient objects can lead to a type lattice. DISIMA distinguishes two kinds of salient objects: physical and logical salient objects. A *logical salient object* is an abstraction of a salient object that is relevant to some application. For example, an object may be created as an instance of type *Politician* to represent President Clinton. The object "Clinton" is created and exists even if there is yet no image in the database in which President Clinton appears. This is called a *logical salient object*; it maintains the generic information that might be stored about this object of interest (e.g., name, position, spouse).

3 MOQL: A MULTIMEDIA EXTENSION OF OQL

OQL allows users to query objects by using their names as entry points into a database. As an embedded language, OQL allows applications to query objects that are supported by the native programming language. The basic statement of OQL is:

select [**distinct**] projection_attributes
from query [[**as**] identifier] {, query [[**as**] identifier] }
[**where** query] [**group by** partition_attributes] [**having** query]
[**order by** sort_criterion {, sort_criterion}]

Most of the extensions introduced to OQL are in the **where** clause in the form of four new predicate expressions: *spatial_expression*, *temporal_expression*, *contains_predicate* and *similarity_predicate*. The *spatial_expression* is a spatial extension which includes spatial objects, spatial functions, and spatial predicates. The *temporal_expression* deals with temporal objects, functions, and predicates. The *contains_predicate* is defined as:

contains_predicate ::= *media_object* **contains** *salientObject*

where, *media_object* represents an instance of a particular medium type, e.g., an image or video object, while *salientObject* is an object within the *media_object* that is deemed interesting (salient) to the application (e.g., a person, a car or a house in an image). The *similarity* predicate checks if two *media_objects* are similar with respect to some metric.

4 VISUALMOQL: THE DISIMA VISUAL QUERY LANGUAGE

The VisualMOQL window consists of a number of components to design a query. The startup window consists of the following components:

- A chooser to select the image classes. Images stored in the database are categorized into user-defined classes. By doing this, the system allows the user to select a subset of the database to search over. The root image class is set as default.
- A salient object class browser which allows the user to choose the objects that he wants. All salient objects and their associated attribute values are identified during the database population time. These objects are organized into a salient object hierarchy with the root salient object class set as the default root in the displayed hierarchy.
- A horizontal slider to specify the maximum number of images that will be returned as the result of the query. This is a quality of service parameter used by the query result presentation interface and not translated into MOQL.
- A horizontal slider to specify the similarity threshold between the query image and the target images stored in the database. It is also used for color comparison.
- A working canvas where the user can construct or modify simple queries.
- A query canvas where the user can construct compound queries based on simple queries using AND, OR and NOT operators.

The user specifies a query by choosing the image class he wants to query and the salient objects he wants to see in the images. Several levels of refinement are offered depending on the type of query and also on the level of precision the user wants the result of the query to have.

4.1 Working canvas

The working canvas is where the user constructs or modifies simple queries. The user can choose either the cognition-based facilities (query by example), referred to as query by sample image, or construct the simple query using the semantic and textual approach, referred to as query by drawing. For the query by drawing approach, the user selects a salient object class in the salient object class browser. The user then draws a rectangle to represent that object on the canvas. This rectangle is used only for determining the spatial relationships between objects. The user can also define the color, shape, texture, and other attribute values of any objects on the canvas by using a dialog box. VisualMOQL allows the user compare textual attributes. Since the variables used to refer to objects in the MOQL translation are shown on the object icons, they can be used to express join operators.

Topological relationships will be deduced for any intersecting objects automatically. Directional relationships have to be defined explicitly through a dialog box. The users specifies which axes (x-axis and/or y-axis) matter. The centroids of the rectangles representing salient objects are used to calculate the directional relationships.

4.2 Query canvas

The query canvas is the space for the user to construct compound image queries. Each simple query is represented by a square box on the query canvas. Compound queries are constructed by combining simple queries or smaller compound queries using AND, OR and NOT operators. A simple query in the query canvas can be modified and revalidated at any stage by using the 'edit' button. This moves the simple query to the working canvas. In order to query over image properties like name of the photographer or the time the image was taken, a dialog box is provided to let user to enter such information.

Finally, the user presses the query button to submit the query. Before translating the visual query constructed by the user, the system will check the query canvas to make sure there are no dangling queries. After this, it will translate the VisualMOQL query into MOQL and display the resulting string before submitting it to the query processor. Figure 1 shows a MOQL query expressing the query Q: “Find images with Bill Clinton dressed in black next to Jean

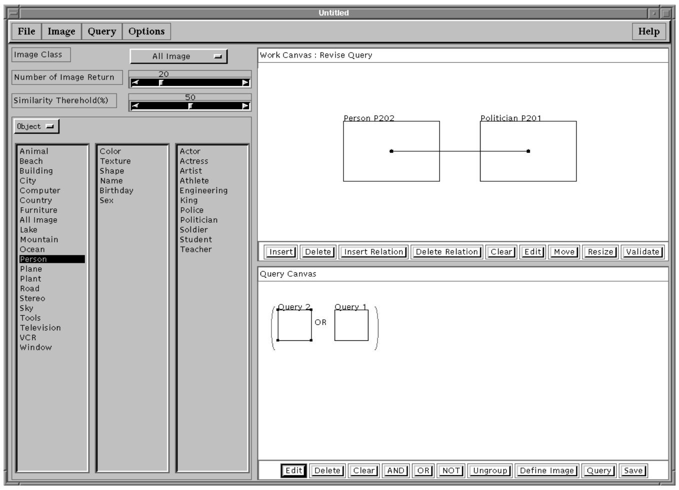

Figure 1 VisualMOQL for Query Q.

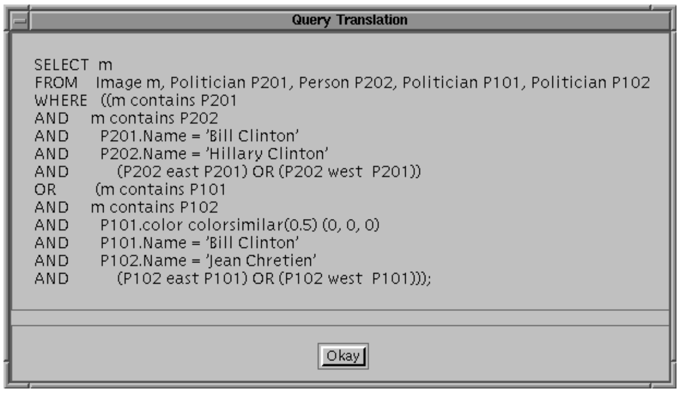

Figure 2 Translated MOQL for Query Q.

Chretien or images with Hillary Clinton sitting next to Jean Chretien". The dialog boxes used for textual attributes and color are not shown. The query is based on a schema corresponding to a news application. The generated MOQL query is given in Figure 2.

5 CONCLUSION

We argue that powerful query languages significantly help simplify multimedia database access. These languages must provide constructs for querying, based on the structure of multimedia data.

In this paper, we have presented VisualMOQL, a visual query language that implements the image features of MOQL. VisualMOQL combines several approaches: cognition-based (query by example), semantic-based (query image semantics) and textual-based (specify and compare attribute values) approaches. A query specified using VisualMOQL is translated into MOQL before execution. As a visual language, VisualMOQL is easy to use but provides the same query facilities as MOQL at the image level.

6 ACKNOWLEDGEMENTS

This research is supported by a strategic grant from the National Science and Engineering Research Council (NSERC) of Canada.

REFERENCES

[ACS90] M. Angelacio, T. Catarci, and G. Santucci. QBD a graphical query language with recursion. *IEEE Transactions on Software Engineering*, 16(10):1150—1163, 1990.

[CBB+97] R. G. G. Cattell, D. Barry, D. Bartels, M. Berler, J. Eastman, S. Gamerman, D. Jordan, A. Springer, H. Strickland, and D. Wade, editors. *The Object Database Standard: ODMG 2.0.* Morgan Kaufmann, San Francisco, CA, 1997.

[LLOW91] C. Lamb, G. Landis, J. Orenstein, and D. Weinreb. The ObjectStore database system. *Communications of ACM*, 34(10):19—20, 1991.

[LÖSO97] J. Z. Li, M. T. Özsu, D. Szafron, and V. Oria. MOQL: A multimedia object query language. In *Proceedings of the 3rd International Workshop on Multimedia Information Systems*, pages 19—28, Como, Italy, September 1997.

[OÖL+97] V. Oria, M. T. Özsu, X. Li, L. Liu, J. Li, Y. Niu, and P. J. Iglinski. Modeling images for content-based queries: The DISIMA approach. In *Proceedings of 2nd International Conference of Visual Information Systems*, pages 239—346, San Diego, California, December 1997.

- Vincent Oria got his Ph.D in Computing Science from the Ecole Nationale Supérieure des Télécommunications, Paris. He is currently a Reasearch Associate at the University of Alberta.
- Bing Xu is finishing his M.Sc, Computing Science at the University of Alberta.
- M. Tamer Özsu is a Professor of Computing Science at the University of Alberta where he leads research groups in distributed object management and multimedia data management.