A Hypertext Query Language for Images

Li Yang*

Institute of Software, Academia Sinica, P. O. Box 8718, Beijing 100080, China

Abstract

HYPERQUERY is a hypertext query language for object-oriented pictorial database systems. First, we discuss object calculus based on term rewriting. Then, example queries are used to illustrate language facilities. This query language has been designed with a flavor similar to QBE as the highly non-procedural and conversational language for object-oriented pictorial database management system OISDBS.

1 Introduction

Most of database management systems that have been implemented to manage pictorial information were developed on top of conventional relational DBMSs. In such systems, an image is represented as physical image(the actual image data) and logical image(image attributes). The logical image, some of which is extracted from physical image by using image processing algorithms, acts as the proxy of physical image in the relational database and provides the only means to access the image data in physical image storage. Obviously, this approach has no modeling ability for physical image data.

This discrepancy of relational pictorial database systems stimulates our motivation of developing an object-oriented image database system OISDBS. By providing formally clear mechanisms for the combination of data structures and operations in abstract data types, OISDBS provides the possibility of image data modeling by describing the structures and interrelationships of image entities. In this paper, we discuss object calculus by using rewrite expressions, and propose a hypertext query language HYPERQUERY which is a query language with the flavor of QBE[2] and QPE (Query by Pictorial Example)[4] but using hypertext forms instead of tables. Following a brief description of object calculus, the language facilities of HYPERQUERY are illustrated through several examples.

*Current address: Department of Intelligence Systems, Institute of Automation, P. O. Box 2728, Beijing 100080, P. R. China

2 Object calculus

To turn to the discussion of object query language, we need a framework of object calculus. Bancilhon *et al* [1] have proposed an object calculus for untyped objects. In this section, we first define a partial order on objects of a type, and show that this partial order induces a lattice structure on the type which allows us to define union and intersection of two objects. Upon these two operations, we can define variables, rewrite rules and rewrite expressions which form the object calculus framework.

Definition 1 *Let O, O' be objects of type R, we can define the fact that O is a* sub-object *of O' (denoted by $O \leq O'$) recursively as follows:*

1. *If O and O' are tuple objects, $O = [O_1, \ldots, O_m]$, $O' = [O'_1, \ldots, O'_m]$, then O is a sub-object of O' if every O_i is a sub-object of O'_i for $1 \leq i \leq m$.*
2. *If O and O' are set objects, $O = \{O_1, \ldots, O_m\}$, $O' = \{O'_1, \ldots, O'_n\}$, then O is a sub-object of O' if every element of O is a sub-object of a corresponding element of O' and there are no distinct objects O_i and O_j in O such that $O_i \leq O_j$.*
3. *Every object is a sub-object of $\top$ and $\bot$ is a sub-object of every object.*

It is clear that sub-object relationship is reflexive ($O \leq O$), transitive (if $O_1 \leq O_2$, $O_2 \leq O_3$, then $O_1 \leq O_3$), and anti-symmetric (if $O_1 \leq O_2$, $O_2 \leq O_1$, then $O_1 = O_2$). For a set of objects of type R, the sub-object relationship $\leq$ is a partial order. Obviously, the following theorem holds:

Theorem 1 *The set of objects of type R, together with the sub-object relationship $\leq$, is a complete lattice.*

With this lattice structure, two basic operations, the union and the intersection of objects, can be defined as follows:

Definition 2 *The union of two objects of a type, $O_1 \cup O_2$, is the smallest object that contains both of them(their least upper bound). The intersection, $O_1 \cap O_2$, is the largest object that is contained in both of them(their greatest lower bound).*

With these two operations, object query can be expressed as rewrite expressions. To define rewrite expressions formally, we first introduce the term *variable*. Variable which has the hierarchical structure as object, is the reflection of object structure in object calculus.

Definition 3 *Given a type R, there exists an infinite set of symbols Var_R, $R \neq R' \rightarrow Var_R \cap Var_{R'} = \phi$, Let $Var = \bigcup_R Var_R$ denotes the set of all symbols,* Variables *are defined recursively as follows:*

1. *Each object of type T is a variable of type T.*
2. *Each element of Var_T is a variable of type T.*
3. *Let $T = [T_1, \ldots, T_n]$ be a tuple type, X_i is a variable of type T_i for $1 \leq i \leq n$, then $[X_1, \ldots, X_n]$ is a variable of type T.*
4. *Let $T = \langle T_1, \ldots, T_n \rangle$ be a union type(every object of type T_i $(1 \leq i \leq n)$ is an object of type T), X_i is a variable of type T_i, $1 \leq i \leq n$, then $\langle X_i \rangle$ is a variable of type T.*
5. *Let $T = \{T'\}$ is a set type, X_i is a variable of type T' for $1 \leq i \leq n$, then $\{X_1, \ldots, X_n\}$ is a variable of type T.*
6. *If X is a variable of type T, then $\geq X, \leq X$, $> X, < X, \neg X$ are also variables of type T.*

The semantics of variables is embodied by the assignment of variables to objects.

Definition 4 *An assignment σ is a mapping which maps variables into objects of appropriate types in the following way:*

1. $\sigma(O) = O$ *for each object.*
2. $\sigma([X_1, \ldots, X_n]) = [\sigma(X_1), \ldots, \sigma(X_n)]$
3. $\sigma(\langle X \rangle) = \langle \sigma(X) \rangle$
4. $\sigma(\{X_1, \ldots, X_n\}) = \{\sigma(X_1), \ldots, \sigma(X_n)\}$
5. *Let X be a variable of type T, then each assignment of $\sigma(\geq X)$, $\sigma(\leq X)$, $\sigma(> X)$, $\sigma(< X)$, $\sigma(\neg X)$ is an object O of type T which satisfies $O \geq \sigma(X)$, $O \leq \sigma(X)$, $O > \sigma(X)$, $O < \sigma(X)$, $O \neq \sigma(X)$ respectively.*

Because variables have the hierarchical structure similar to objects, the assignment of variables can be a part of query language. Consider a variable V of type T with variables $V_1, \ldots, V_n$ in it, and an assignment σ of objects $O_1, \ldots, O_n$ to variables $V_1, \ldots, V_n$, σ will also assign an object of type T to the variable V. More generally, we have:

Definition 5 *Let V be a variable of type T, O an object of T, the effect of V on O is a set which is the union of all assignments of V that are sub-object of O.*

$$V(O) = \bigcup \{\sigma(V) | \sigma(V) \leq O\}$$

Therefore, the problem with variables to represent a query is that it only selects a set of sub-objects and not allow object restructuring. To overcome this, we introduce the notion of rewrite rules and rewrite expressions.

Definition 6 *Let X be a variable of type S, rewrite rules and rewrite expressions are defined recursively in the following way.*

1. *Let Y be a variable of type T,*
   (a) *$X \rightarrow Y$ is a rule from S to T.*
   (b) *if ρ is a rewrite expression from T to T', $X \rightarrow \rho(Y)$ is a rule from S to T'.*
2. (a) *if $X \rightarrow Y_1, \ldots, X \rightarrow Y_n$ are rules from S to $T_1, \ldots, T_n$ respectively, then $X \rightarrow [Y_1, \ldots, Y_n]$ is a rule from S to $[T_1, \ldots, T_n]$.*
   (b) *if $X \rightarrow Y$ is a rule from S to some T_i in $T_1, \ldots, T_n$. then $X \rightarrow \langle Y \rangle$ is a rule from S to $\langle T_1, \ldots, T_n \rangle$.*
   (c) *if $X \rightarrow Y_1, \ldots, X \rightarrow Y_n$ are rules from S to T, then $X \rightarrow \{Y_1, \ldots, Y_n\}$ is a rule from S to $\{T\}$.*
   (d) *if $X \rightarrow Y_1, \ldots, X \rightarrow Y_n$ are rules from S to $\{T\}$, then $X \rightarrow Y_1 \cup \ldots \cup Y_n$ is a rule from S to $\{T\}$.*
3. *a rewrite expression from $\{S\}$ to $\{T\}$ has the form* rew(Δ), *where Δ is a set of rules from S to T.*

In Definition 4, we have defined the assignment of variables to objects of appropriate types, and indicated that it can be a part of object query language. In the following, we will indicate that this kind of assignment can be extended to the assignment of rewrite expressions which forms the entire object calculus framework.

Definition 7 *Let σ be an assignment of variables, then*

1. $\sigma(O) = O$ *for each object* O.
2. $\sigma([X_1, \ldots, X_n]) = [\sigma(X_1), \ldots, \sigma(X_n)]$
3. $\sigma(\langle X \rangle) = \langle \sigma(X) \rangle$
4. $\sigma(\{X_1, \ldots, X_n\}) = \{\sigma(X_1), \ldots, \sigma(X_n)\}$
5. $\sigma(X_1 \cup \ldots \cup X_n) = \sigma(X_1) \cup \ldots \cup \sigma(X_n)$
6. $\sigma(\text{rew}(\Delta)(X)) = \{\beta$
   $\sigma(X)\}$

$(V)|U \rightarrow V \in \Delta, \beta(U) \in$

The semantics of rewrite expressions can be defined as follows:

Definition 8 *Let $\rho = \text{rew}(\Delta)$ be a rewrite expression for type S, O an object of S, then the effect of ρ on O is defined as*

$$\rho(O) = \{\sigma(Y)|X \rightarrow Y \in \Delta, \text{and } \sigma(X) \leq O\}$$

3 The language

Obviously, rewrite rules and rewrite expressions can be used as a kind of query language, but it is not very user friendly. To meet the requirements for user friendness, rewrite expressions should be converted into more acceptable forms. In relational databases, the tabular query language QBE[2] based on domain relational calculus provides a brief, easy to understand guidance in query formulation. Query by Diagram[3] is a query language directly on the E-R graph. Query by Pictorial Example(QPE)[4] which adopts the QBE approach is a tabular query language for pictorial database system. In the design of HYPERQUERY, we wish to adopt this flavor to express a query by entering an example in the appropriate location on screen.

Compared with QBE and QPE, HYPERQUERY uses hypertext form in the formulation of query instead of tables. It is easy to formulate a query in hypertext form for object-oriented databases because hypertexts are just the user's view of the database. In HYPERQUERY, each operation is specified by using one or more hypertext forms. Each form is built up on screen with the entire structure being supplied by the system and other parts by the user. *Example elements* which are marked underlining are variables specified by the user and solved by the system during query processing.

Query1: Print all forest farms of which the area is above 100 hectares and the managers include a person named WANG.

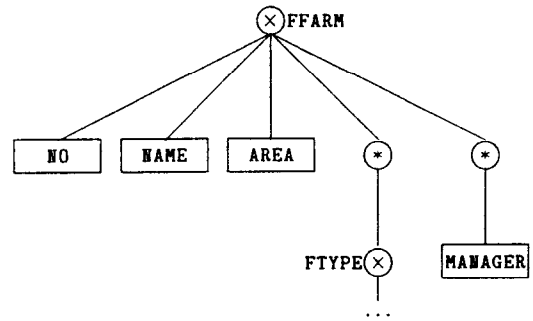

Figure 1: The structure of FFARM supplied by the system in Query 1

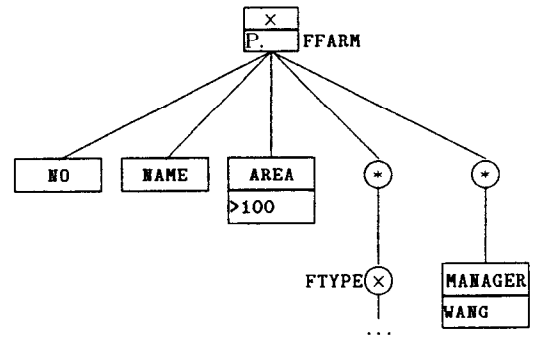

Figure 2: The formulation of Query 1

Initially, the user, knowing the answer to the query is in type FFARM, enters FFARM as the type name, the system will then respond by giving the structure of type FFARM, as in Figure 1.

Once the user enters a type name, the system only gives the structure of a type till the next level ×-vertex in depth. Now the user can express the query on the terminal by moving the cursor to the appropriate position, clicking the mouse button to push out a small window, and filling it. The query is shown in Figure 2.

In Figure 2, "P." stands for "PRINT", an output operator. This query can also be formulated as variables:

[NO:X, NAME:Y, AREA:> 100, {FTYPE:Z},
≥{MANAGER:WANG}]

Query 2: Display all forest types in No. 2 forest farm.

First, the user enters FFARM as the type name, the system then responds the same as in Query 1. Because this query is about the type FTYPE, the

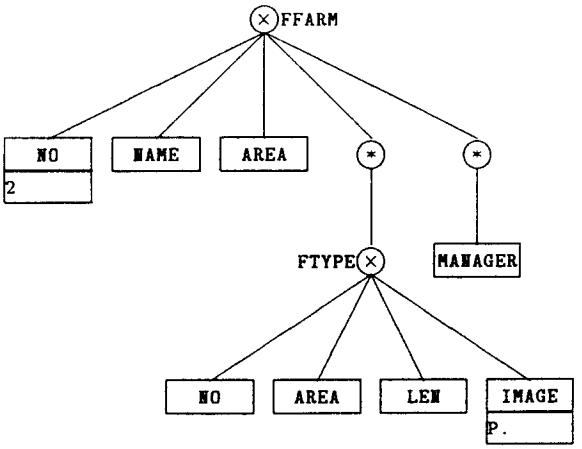

Figure 3: The formulation of Query 2

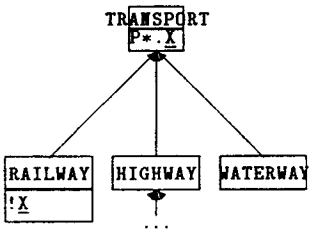

Figure 4: The formulation of Query 3

user moves cursor to the node FTYPE and clicks the button, the structure of type FTYPE will appear. By the appropriate positions being filled, this query can be expressed in Figure 3.

Query 2 can also be formulated in rewrite expression as follows:

rew([NO:2, NAME, AREA,
{FTYPE:[NO, AREA, LEN, IMAGE:X]},
{MANAGER}] — IMAGE:X)

Query 3: Print all transport lines except the railway.

The type TRANSPORT is the generalization of RAILWAY, HIGHWAY and WATERWAY. HIGHWAY is the generalization of types ARTERIAL and BRANCH. When the user specifies the name of a generalized type and an asterisk(*), the system will display all its direct subtypes. In this example, the user enters "TRANSPORT", and the query can be expressed in Figure 4.

The "*" in operator "P*" means that the query is applied to all of the subtypes of TRANSPORT.

Query 4: Print and display all the rivers whose discharge is above 100 cubic meter per second.

As usual, rivers are classified into two kinds, single line rivers and double line rivers, which are stored in the database differently. A single line river, which is perhaps a small river or a stream, is stored as one line in the database while a double line river is stored as two lines which represent two sides of the river. In this example, the type RIVER is the union of types SINGLELINE and DOUBLELINE. To formulate this query, the user must specify DISCHARGE for SINGLELINE and DOUBLELINE separately(Figure 5).

Query 5: To illustrate the query about more than one type and the use of image operators, let us consider another example: In order to know how many bridges are needed in the design of a railway and the type of each bridge, we must know all the rivers across this railway. This query involves two types, RAILWAY and RIVER, and can be expressed as in Figure 6.

In this query, these two types are connected by common *example element* X. In addition, `intersect(X)` is an image operator which determines two line objects intersect or not. In the system, the image part of each pictorial object contains its MBR(Maximum Boundary Rectangle). The `intersect` operator first compares MBRs of these lines rather than performs point-to-point matching directly which is very time-consuming.

4 Implementation issues

HYPERQUERY is designed as the query language for OISDBS (Object-oriented Intelligent Spatial DataBase System). OISDBS is built on ISDBS[5], a relational spatial database system. It adds an Object Manager on top of ISDBS which supports object-oriented database schema design, storage management, indices, and the mapping from object schema to relational schema. OISDBS is currently being implemented on a VAX 8700 under the VMS operating system. The user interacts with the system via a VT340 graphic terminal.

5 Summary

The object-oriented database query language HYPERQUERY is introduced. Object calculus based on term rewriting is also discussed. In query formulation, queries about pictorial entities can be expressed just as the user's view of the object database. Pictorial operations are introduced for the manipulation regarding pictorial entities. Queries about a type can be expressed as a kind of hypertext, while queries about more than one type can be formulated by introducing *example elements*. All these features make HYPERQUERY a very attractive and versatile query

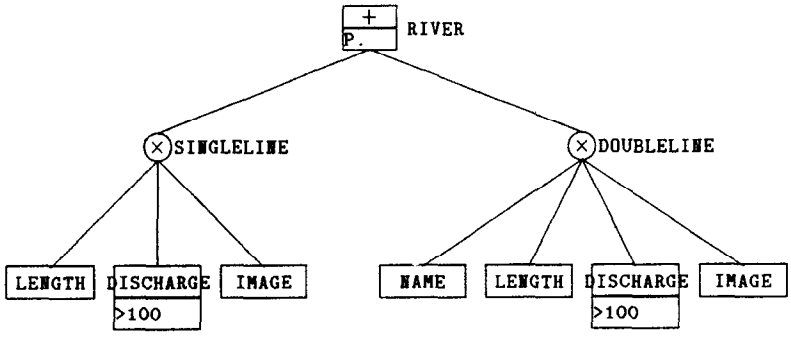

Figure 5: The formulation of Query 4

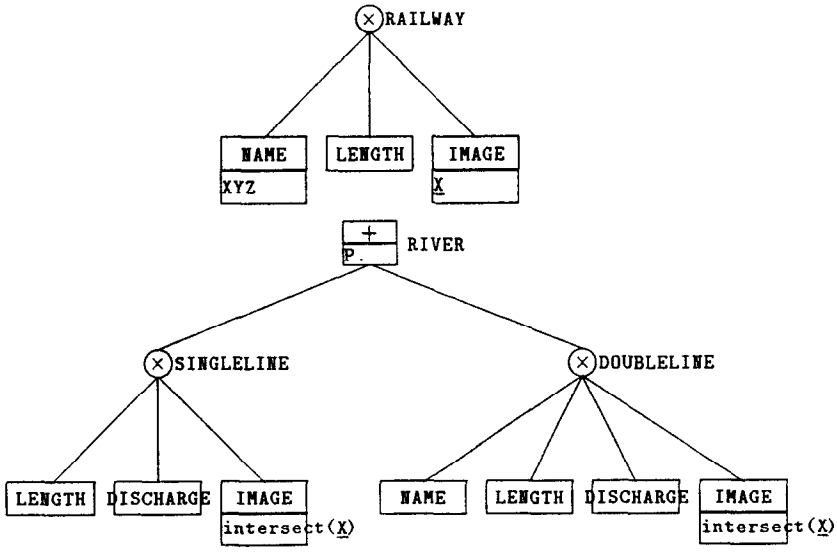

Figure 6: The formulation of Query 5

language for the manipulation of pictures by decision makers, resource managers, and image processing engineers.

A subset of HYPERQUERY has been illustrated through examples. It is clear that HYPERQUERY approach is a rather open ended language which can be extended to accommodate additional or new functions that may be of particular interest to specific areas. It provides a basis upon which further language operations for pictorial data may be easily built.

References

[1] F. Bancilhon, S. Khoshafian, A calculus for complex objects, in *Proc. 5th ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, Cambridge, Mass., Mar. 1986, 53–59

[2] M. Zloof, Query-by-example: a database language, *IBM System Journal*, **16**, 1977, 324–343

[3] G. Santucci, P. A. Sottile, Query by Diagram: a visual environment for querying databases, *Software - Practice and Experience*, **23**(3), 1993, 317-340

[4] N. S. Chang, K. S. Fu, Query-by-pictorial example, *IEEE Trans. on Software Engineering*, SE-**6**(6), 1980, 519–524

[5] J. K. Wu, T. Chen, L. Yang, A knowledge-based image database system(ISDBS), *Science in China*(Series A), **34**(1), 1991, 87-93