A Visual Environment for Multimedia Object Retrieval

Dario Lucarella\textsuperscript{1,2}, Antonella Zanzi\textsuperscript{1}, Mauro Zanzi\textsuperscript{2}

\textsuperscript{1} Centro Ricerca di Automatica-ENEL, Via Volta 1, I-20093 Cologno M. (Milano), Italy
\textsuperscript{2} Università degli Studi di Milano, Via Comelico 39, I-20135 Milano, Italy

Abstract. We present a graph-based object model that has been used as a uniform framework for direct manipulation of multimedia information. After a brief introduction motivating the need for abstraction and structuring mechanisms in hypermedia systems, we introduce the data model and the visual retrieval environment which, combining filtering, browsing and navigation techniques, provides an integrated view of the retrieval problem.

Introduction

Hypermedia systems are oriented to manage a collection of information units that are arbitrarily diverse in form and content. Such units may contain texts, graphics, images, sound, video and animation and are connected by logical links to form an information network.

There is a growing interest today in such technologies for the implementation of massive multimedia information systems, but unfortunately, several well-recognized problems continue to be open research issues \cite{8}. The simplicity of the basic hypermedia model does not appropriately represent the structure of the information \cite{5}. As a result, the user has difficulty in perceiving the conceptual model of the application and a cognitive overhead is required in authoring mode as well as in reading mode.

Recently, the requirements for representing the complex interrelationships that arise in hypermedia have generated a renewed interest in semantic data models \cite{11}. In particular, graph-based data models seem to provide a natural way of handling data appearing in applications such as hypertext or multimedia information systems \cite{4, 7, 2}. In this direction, we propose a graph-based object model and we use it as a uniform framework both for conceptual modeling and for direct manipulation of the stored objects.

As soon as the underlying data model becomes more complex, the level of complexity of the associated query language also increases, and so the main goal becomes the design of a language that provides both high semantic power and ease of interaction. With this objective, we propose a visual retrieval paradigm. The user is not required to know any complex formal language, with the advantage of maintaining the same interaction style normally used during browsing \cite{9}.

Much research is being carried out in the database community on graphical query languages. Two systems that, although based on the E-R model, have influenced our approach are QBD* \cite{1} and SUPER \cite{3}. The visual interface to GOOD \cite{6}, being based on a graph-oriented model, exhibits some features in common with our system.

A graph-based object model

Basic notions of object, class, and abstraction mechanisms are assumed. We give a formal definition of the model:

Definition. The conceptual schema \(\Sigma\) is defined as the five-tuple \(\Sigma = (C, T, A, \mathcal{H}, \mathcal{P})\) where:

\begin{itemize}
  \item \(C\) is a finite set of class names; each class \(c \in C\) denotes a structure (in terms of attributes) and it also denotes an extension (the collection of objects that have that structure).
  \item \(T\) is a finite set of type names (e.g. integer, text, picture, etc.) built-in the system; each \(t \in T\) denotes a type of primitive objects and \(V(t)\) is the set of associated values.
  \item \(A\) is a finite set of attribute names. Attributes may be simple or complex. The domain of a simple attribute is a basic type \(t \in T\); the domain of a complex attribute is a class \(c \in C\). In addition, we distinguish between single-valued attributes \(A_s\) and multi-valued attributes \(A_m\), with \(A = A_s \cup A_m\).
  \item \(\mathcal{P} \subseteq C \times A \times (C \cup T)\) is the property relation. If \((c_i, a, c_j) \in \mathcal{P}\), then the class \(c_i\) has the attribute \(a\), having as a domain the class or type \(c_j\).
  \item \(\mathcal{H} \subseteq C \times C\) is the inheritance partial ordering relation. If \((c_i, c_j) \in \mathcal{H}\), then the class \(c_i\) is a subclass of the class \(c_j\) inheriting attributes from \(c_j\).
\end{itemize}

Definition. Given \(\Sigma\), the conceptual schema graph is a directed labeled graph \(G(\Sigma) = (N, E)\) where:

\begin{itemize}
  \item \(N = C \cup T\) is the set of nodes. For each \(c \in C\), we have a rectangular-shaped node labeled \(c\). For each \(t \in T\), we have an oval-shaped node labeled \(t\).
  \item \(E\) is the set of edges. For each \((c_i, c_j) \in \mathcal{H}\) we have a bold edge connecting \(c_i\) to \(c_j\). For each \((c_i, a, c_j) \in \mathcal{P}\) we have an \(a\)-labeled edge from \(c_i\) to \(c_j\). Particularly, if \(a \in A_s\), we have an edge with a single arrow; if \(a \in A_m\), we have an edge with a double arrow.
\end{itemize}

Definition. The multimedia information system \(M\) is defined as the four-tuple \(M = (\Sigma, O, \mathcal{I}, \mathcal{L})\) where:

\begin{itemize}
  \item \(\Sigma\) is the conceptual schema defined above.
  \item \(O\) is the set of objects stored into the system.
\end{itemize}
– $I \subseteq O \times C$ is the instantiation relationship. Each object $o \in O$ is an instance of a class $c \in C$.

– $L \subseteq O \times A \times (O \cup V(T))$ is the link relationship. If $(o_i, a, o_j) \in L$, then the attribute $a$ of the object $o_i$ has the value $o_j$. Assuming $o_i$ instance of $c_i$ and $o_j$ instance of $c_j$, we have $(o_i, a, o_j) \in L$ if one of the following conditions holds:

(i) $(c_i, a, c_j) \in P$;

(ii) $(c_i, c_k) \in H \land (c_k, a, c_j) \in P$;

(iii) $(c_j, c_k) \in H \land (c_i, a, c_k) \in P$.

The last two conditions are the direct consequence of the semantics of the inheritance relationship.

**Definition.** Given the multimedia information system $M$, an *instance graph* is a directed labeled graph $G(M) = (N,E)$ where:

– $N = O \cup V(T)$ is the set of nodes. Nodes represent objects (rectangular nodes) or values (oval nodes) generated from the schema through the instance relationship;

– $E$ is the set of edges. For each $(o_i, a, o_j) \in L$ there is an $a$-labeled edge from $o_i$ to $o_j$.

**Object retrieval and access**

In analogy with the views in databases, we introduce the notion of *perspective*\(^1\). Essentially, perspectives are graph structures that are built from the schema graph and are visually operated in various ways.

**Definition.** Given a multimedia information system, a *perspective* $P$ is defined as $P(\pi, S)$ where:

– $\pi$ is the perspective *pattern*, i.e. a weakly connected subgraph\(^2\) of the schema graph $\Sigma$; hence $N(\pi) \subseteq N(\Sigma)$ denotes the subset of schema nodes (classes and types) included in the perspective, and $E(\pi) \subseteq E(\Sigma)$ denotes the set of edges (properties) associated with such nodes.

– $S$ is the set of object graphs generated by the perspective graph $\pi$ through the instance relationship. Given an instance $s \in S$, each node $o \in N(s)$ is an instance of the corresponding node $c \in N(\pi)$ and the edge $(o_i, a, o_j) \in E(s)$ iff the edge $(c_i, a, c_j) \in E(\pi)$.

So, a perspective is defined by a pattern (the intentional representation) and by the corresponding object graphs (the extensional representation).

In order to restrict the attention to a subset of pattern instances in the perspective, a *filter* can be defined over it.

**Definition.** Given a perspective $P(\pi, S)$, a *filter* $F$ is defined in terms of a set of selection conditions $\{C_1, \ldots, C_n\}$ over the pattern. Let $a_i$ be an attribute of type $t$ pertaining to a node (class) $c_i$ in the pattern $\pi$, then $C_i$ represents a selection condition over the actual values of the corresponding object instances. The selection condition is a Boolean combination $\land, \lor, \neg$ of simple expressions of the form $(a_i \circ \alpha_j)$ where $\alpha_j$ is a type-compatible property or a *constant*, i.e. some value from the domain $t$; $\circ$ is a comparison operator depending on the type of the operands.

**Definition.** Given a perspective $P(\pi, S)$ and a filter $F$ defined over $P$, a *selection operation* $\sigma$ returns a subset $R \subseteq S$ of pattern instances matching the filter:

$$\sigma_F(P) = R = \{s \mid s \in S \land s \models F\}$$

A pattern instance $s$ matches the filter if and only if it satisfies all the conditions $\land_{i=1,n} C_i$; a condition $C_i$ over the class $s$ is satisfied if it is true for the corresponding object instance values.

**Definition.** Two perspectives $P_1(\pi_1, S_1)$ and $P_2(\pi_2, S_2)$ are said to be *compatible* when they have the same pattern, but different instance sets: $\pi_1 = \pi_2; S_1 \neq S_2$. This is the case resulting from the application of different filters to the same original perspective. Now we introduce binary operations to combine perspectives together.

**Definition.** Let $P_1(\pi_1, S_1)$ and $P_2(\pi_2, S_2)$ be two perspectives, with $\pi_1 \neq \pi_2$ and $N(\pi_1) \cap N(\pi_2) \neq \Phi$, a *composition* operation $\odot$ over the set of nodes $N' = N(\pi_1) \cap N(\pi_2)$ generates the perspective $P(\pi, S) = P_1 \odot P_2$ where:

– $\pi$ is the composition of the two patterns $\pi_1$ and $\pi_2$ obtained by taking the union of nodes and edges respectively, $N(\pi) = N(\pi_1) \cup N(\pi_2)$ and $E(\pi) = E(\pi_1) \cup E(\pi_2)$;

– $S$ is the set of instances of the pattern $\pi$ obtained by composing the instances in $S_1$ with those in $S_2$; two instances are composed iff for all the nodes (classes) $N' = N(\pi_1) \cap N(\pi_2)$, they share the same object instance.

This last operation is effective in getting pattern instances satisfying a disjunction of conditions.

The composition and overlay operations are very useful in combining perspectives together. Just to give an idea of the user interface environment, in figure 1, an actual interaction session is reported concerning the manipulation of perspectives.

In order to access and view the objects in the instantiation of the perspective, *browsing* and *navigation* operations are available.

**Definition.** Given a perspective $P(\pi, S)$; let $c \in N(\pi)$ be a node (class), a *browsing* operation $B$ returns all the relative object instances iff included in one of the pattern instances $s \in S$:

$$B_c(P) = \{o \mid o = I(c) \land o \in N(s)\}$$
By default, simple attributes are embedded into the window with layout (e.g. font, color, size, etc.) derived from information set up during the loading. Conversely, complex attributes are depicted in the window as “icon buttons” that can then be activated in navigation mode.

**Definition.** Given a perspective $P(\pi, S)$; let $o$ be a displayed object in the instance $s \in S$, and let $a$ be one of its complex attributes, then a navigation operation $N$ returns the linked objects:

$$N_a(o)(P) = \{ o' | (o, a, o') \in \mathcal{L} \land o' \in N(s) \}$$

Note that the last condition in the definition prevents the user from accessing nodes outside of the perspective. This is achieved dynamically by disabling those buttons pointing to objects out of the perspective.

**Conclusions**

We have presented a graph-based object model which has been used as a uniform framework for visual information retrieval. These concepts have been demonstrated in the implementation of the MORE prototype system [10], which has been tested with reference to a significant hypermedia application. The considerations developed and the first experiments carried out in this actual setting suggest that the combination of a graph-based object model with the direct manipulation paradigm provides great flexibility for conceptual modeling, as well as for the effective retrieval of multimedia information.

**References**


