Object-Oriented Modeling and Querying of Hypermedia Documents

Georges Gardarin Shim Yoon
PRiSM Laboratory
University of Versailles Saint-Quentin
45 Avenue des Etats-Unis
78035 Versailles Cedex, France
e-mail: {Georges.Gardarin, Shim Yoon}@prism.uvsq.fr

Abstract

This paper presents an object model of hypermedia documents and proposes a language to query them. The main idea is to use standard concepts of OODB such as encapsulation, relationships, inheritance, and path traversal. Documents, composed by using the HyTime standard, are transformed and stored in a hypermedia OODB using an HyTime interpreter. They can then be retrieved using extensions of the OQL query language not only by content-based but also by structure-based queries. Especially the inheritance mechanism is used between application classes and the system classes where common attributes and operations are defined to provide the capability of structural queries. We also propose additional query features to extend the quantifiers of the OQL language for hyperdocuments. The ODBMS O2 is used to implement a prototype of the proposed system.

1 Introduction

An important trend in database evolution is the development of applications to store and retrieve multimedia documents more effectively. Among these applications, the combination of hypertext and structured multimedia documents giving hypermedia documents is of special interest. Hypermedia as well as hypertext [5] is becoming very popular as an approach to manipulate multimedia documents in a non-linear way. In these systems, data are organized in a network of text or multimedia data nodes connected by links. Many of these systems embody a labelled directed graph model in which nodes and edges correspond respectively to document contents and links among them. The retrieval is done by navigation: users traverse links in document networks by pressing a link button on the screen. The focus is not put on the content structure of a document but on the global structure of a document network.

The domain of structured documents has been developed in a manner that preserves the document hierarchical structure: a higher-level document object is formed from lower-level objects. Documents are represented by the description of their schema often mixed with the description of their content. Data models and retrieval operations have been proposed to support these representations and to manipulate documents. Clifton [4] suggested a model in which an object consists of a set of triples composed by type, key, and data item, where the types define the schema of the database. Güting [10] proposed a model based on a nested sequence of tuples, where a data object is a pair composed of a schema and an occurrence. He also described an algebra to retrieve and manipulate such objects.

To exchange documents between different information systems, the standardization organizations have normalised document structures. ODA (Office Document Architecture) is defined in the ISO 8613 standard; it models the representation of documents and their exchange. SGML (Standard Generalised Markup Language) [20] is defined in the ISO/IEC 8879-1986 standard; it specifies a declarative language for the representation of documents. It uses a markup for describing document structures. The HyTime standard [12] is an application of SGML for providing means of
expressing hypermedia information using a standard language. HyTime is a language to describe and constitute not only structured documents but also link structures.

The development of HyTime makes possible the integration of two document processing domains: hypertext and structured document. We study the similarities and differences between these two domains and point out the lacks of each field to realize their integration in a database context. On the one hand, an appropriate data model is required for integrating both structured and hypertext document. On the other hand, a powerful query mechanism is required for providing both content-based and structure-based access. Navigational access in hypertext systems has proven to be insufficient when there are too many nodes in a network. Query-based access mechanisms are limited to retrieving the nodes and links based on predicates referring attributes. As Halasz [11] pointed out for hypertext databases, a query language able to retrieve parts of a hypermedia based on structural specifications is required.

The remainder of this paper is organized as follows. In Section 2, we present the architecture of the system prototype. Section 3 outlines the proposed model for hypermedia documents. Section 4 presents the query language HyOQL, which extends OQL to query HyTime documents. Section 5 concludes the paper.

2 Architecture of the System

This section gives a brief sketch of the system architecture that we are currently implementing. Figure 1 shows the procedural architecture.

While much research has been conducted on modeling hypertext documents [7, 15, 16, 17], little progress has been made toward integrating hypertext and structured multimedia objects [9, 13]. In the domain of hypertext querying, graphical query languages such as G+ [8] and GraphLog [6] permit to formulate a query by drawing graph patterns. While these languages have suggested effective solutions to specify any subsets of the network with the document structures, mixing content and structure in queries is still difficult. Beeri and Koranatzky [5] have investigated a logical query language that allows both content-based and structure-based queries, supporting a rich set of quantifiers such as most, many and few. Afrati and Koutras [1] have defined a hypertext model and proposed a Datalog-like query language on this model. An algebraic query language has been proposed by Amann [2]. While these researches support more complete query facilities, they are yet short of user-friendliness.

The goal of this paper is to present a data model and a query language that deal with hypertextdocuments integrating multimedia objects within a hypertext-like network. We base our approach on the HyTime standard. Thus, hyperdocument means two or more documents or other information objects that are interconnected by a set of one or more hyperlinks [12]. We use standard concepts of object-oriented modeling such as class hierarchy, inheritance, and reference attribute to design the schema of the hypermedia databases. This approach allows the documents to be retrieved and processed efficiently in a database-oriented fashion. As far as queries are concerned, we present a query language permitting both content-based and structural queries. Our query language named HyOQL is an extension of the standard OQL query language proposed by the ODMG [19]. We extend OQL by adding special operations necessary to express structural hypertext queries more naturally and by providing a richer set of quantifiers.

The goal of this paper is to present a data model and a query language that deal with hypertextdocuments integrating multimedia objects within a hypertext-like network. We base our approach on the HyTime standard. Thus, hyperdocument means two or more documents or other information objects that are interconnected by a set of one or more hyperlinks [12]. We use standard concepts of object-oriented modeling such as class hierarchy, inheritance, and reference attribute to design the schema of the hypermedia databases. This approach allows the documents to be retrieved and processed efficiently in a database-oriented fashion. As far as queries are concerned, we present a query language permitting both content-based and structural queries. Our query language named HyOQL is an extension of the standard OQL query language proposed by the ODMG [19]. We extend OQL by adding special operations necessary to express structural hypertext queries more naturally and by providing a richer set of quantifiers.

The remainder of this paper is organized as follows. In Section 2, we present the architecture of the system prototype. Section 3 outlines the proposed model for hypermedia documents. Section 4 presents the query language HyOQL, which extends OQL to query HyTime documents. Section 5 concludes the paper.

2 Architecture of the System

Figure 1: Procedural architecture of the prototype

HyTime documents are composed of two types of documents, i.e., a DTD (Document Type Definition) document that specifies the structure of the document and a DOC document that contains the actual content according to this structure. These documents are sequential files edited with any word processors or editors. The HyTime parser analyses first the HyTime DTD document to generate the corresponding document classes. Then, it transforms and stores the real documents into the document class extents. After the transformation, the documents become a set of objects in the database. We provide predefined system classes called base classes factoring common attributes and operations. The goal of our object-oriented modeling is to define these attributes and operations so that they can be used not only in queries but also in application programs, thus providing hypermedia functionalities through persistent object-oriented languages.
Figure 2 represents a functional architecture of the prototype. The Document Classes are the classes generated by the parser from the DTD of the HyTime documents. They correspond to application specific concepts, e.g., documents, papers, sections, and paragraphs. Users can also define the document classes directly in the DBMS independently of the HyTime specification. The Base Classes are the results of our document modeling.

The Library provides functionalities that are specific to hypermedia applications such as a set of graphic interface functions intended to support hypermedia structures. As an example, utility functions to display an overview of a hypermedia network can be defined in this library. This component is beyond the scope of this paper. We use the O2 OO DBMS as a storage system to develop the prototype. Unlike other systems that use DBMS only for storing link structures and that stores documents in ordinary files, our prototype stores all the documents and the internal structures in the DBMS. It allows us to benefit the DBMS functionalities.

3 Hypermedia Document Modeling

This section introduces the data model proposed for hypermedia documents. We consider the two aspects of HyTime documents: hypertext structures and hierarchical document structures. A hypertext structure can be viewed as a network of nodes which contain information and are connected by links. In HyTime, the linking capability is supported in the Hyperlink module. In our model, this hypertext structure is supported by the HyLink and HyAnchor classes. A hierarchical document structure is supported by the basic functionality of SGML: each document consists of parts which in turn are composed of sub-parts and so on down to atomic units. This structure is supported by an object-oriented data model which allows to define and manipulate complex objects. For example, the complex object, Paper, can be defined by applying constructors such as Set, List, Bag, and Array. A paper consisting of a list of sections can be defined as having an attribute referencing the section objects. In order to easily access complex objects, OO DBMS provides path traversal facilities.

3.1 Simple example

As a simple example, we consider an electronic paper reviewing system. A paper consists of a title, authors, photos of the authors, an abstract, and a paper body. The abstract consists of paragraphs. The paper body consists of sections, and a section consists of paragraphs. Finally a paragraph contains the actual text (See Figure 3). Figure 4 represents a hypermedia structure for an paper reviewing application. A paper can refer to another paper or notes; various comments from readers can refer to it in a positive or negative way. A comment may itself be commented.

Figure 3: Hierarchical schema for a typical paper

Figure 4: Hypertext schema for papers

Figure 5 shows a simplified version of a DTD for a paper document. For simplicity, we omit some declarations such as HyTime support, HyTime version, and HyTime module. DTD files specify the name of elements and the structures of these elements by using the keyword !element. They specify the attributes of each element by using the keyword !attlist. The element clink is a predefined element for specifying link references. The expression +(clink) in the first line of the DTD means that the element clink can be used several times in the element Paper. More generally, the occurrence symbol + is used to say that an element can be multiple (e.g., para in abstract). (#PCDATA) is a keyword of SGML to define any characters.

In figure 6, we give an instance of a paper document written according to the structure defined in the DTD of Figure 5. It uses "markup" consisting of a start tag (<Paper>) and optionally an end tag (<</Paper>) to indicate an element of a document. The attributes defined in the DTD appear in the start tag (<Paper id=P1>). The paper is identified by the attribute id of value "P1". The first link is defined for the authors' notes to the destination node NoteAuthor and the second link is for a referenced paper on HyTime.
The texts between the start tag and the end tag of the link may be represented differently on the screen in order to show that these are the sensitive zones on which the user can click to reach other documents.

3.2 Class Hierarchy

As mentioned in the architecture section, a document model is derived from the class hierarchy of the base classes. Figure 7 shows the base class hierarchy. We use the object diagram of OMT(Object Modeling Technique), more precisely a class diagram, to model classes and their relationships. A rectangular box represents a class with its attributes and optionally its operations. Solid lines represent relationships among classes. The solid balls mean multi-valued relationships. And the number at the end of a solid line specifies the number of instances. The diamond symbol indicates an aggregation that is the "a-part-of" relationship. The triangle symbols are used to represent inheritance or generalization between a superclass and its subclasses.

The class HyNode is the root class from which the document classes inherit properties. This class models a node consisting of outgoing and incoming links to/from another node in a hypermedia document network. Almost all of the operations supporting queries are implemented in this class.

3.3 Schema of the Base Classes

We now give a more detailed description of each base class defined using ODL (Object Definition Language) of ODMG group. ODL is a specification language used to define the characteristics of types, including their properties and operations. In the ODMG object
model, a type defines the properties and operations of its instances. A type has one or more implementations and the combination of type specification and one of the implementation is named a class. Since our schema is defined using ODL, it can be supported by all ODMG-compliant ODBMS.

The HyDoc class definition is given in Figure 8. This class, identified by the docid attribute, represents a network of hyperdocuments as a set of HyNode objects linked by a set of HyLink objects. We defined operations for adding and deleting a node or a link in this network. The subgraph operation creates a new HyDoc object from a list of sets of nodes. It is initially defined to contain the result of a query. The system generates a new docid by adding a prefix to the initial docid. The graph operation filters the HyDoc network to search a sub-network satisfying the regular expression given in the argument path as a string.

The class HyNode is shown in Figure 9. Each HyNode object is identified by its nodeid attribute. The relationship Is-nodein referencing a HyDoc object indicates to which HyDoc network it belongs. The outgoing and incoming links are modeled by the two relationships Has-inlinks and Has-outlinks, which are lists of HyLink objects. Operations are defined to create, delete, modify, and display a HyNode object. Forwnode and forwlink operations follow a link given as a parameter and results in the corresponding node or link respectively.

We implement a series of operations to get the children or the parents of a given node. The forwnodes operation gives all the nodes that leave from a node and backnodes gives all the nodes that enter into a node satisfying a given role, i.e., a link label, when specified. Forwlinks and backlinks do the same thing except that they return a set of links instead of nodes. The two operations forwpath and backpath find the descendant and ancestor nodes by traversing the links up to a given level. If the level parameter is not given, the transitive closure of the network is computed.

The HyLink is defined in Figure 10. A HyLink class identified by the linkid attribute is the general class for hyperlinking capability. It has the relationship Is-linkin representing which document network this link belongs to. The two relationships Is-inlinkin and Is-outlinkin reference the node where this link is incoming and outgoing respectively. The attribute role corresponds to a link label. There are four internal operations to create, delete, modify, and display a link. Clink and ilink are subclasses of HyLink. They have the Has-clinkends and Has-ilinkends relationships that point to the HyAnchor class. Note that the clink has only two linkends.

The HyAnchor is shown in Figure 11. The HyAnchor identified by the anchorid attribute is the general anchor class. It has two relationships Is-clinkendin and Is-ilinkendin that represent which clink and ilink this anchor belongs to. As mentioned before, three kinds of anchors are defined in our model. The first is an icon that represents a graphical image by the icon attribute. The x and y attributes indicate the location where this icon is displayed on the screen. The second is a region, that is a portion of a text. The four attributes in the AnchorRegion class are used to define a portion in a text. The third type of an anchor is a node. An entire node can be an anchor. AnchorNode class is a subclass of the two classes HyAnchor and HyNode, hence it can take advantage of multiple inheritance. Finally, Figure 12 defines the class Paper generated from the HyTime document by the parser. This class inherits from the predefined class

---

2Strictly speaking, this is a type definition in the ODMG model, but we will use the term class, assuming that each type has at least one operation implementation.

3We used the indexed List type for relationships, though this is not yet defined in ODL.
HyNode. Thus, the class Paper can use all the operations defined in the class HyNode. Note that not all the HyTime elements are transformed into classes. The choice whether the para element is transformed into a class or not is made by the parser. Transformation rules are needed to make the decision.

```java
interface HyLink {
    extent hylinks; key linkid;
    attribute String linkid;
    attribute String role;
    relationship HyDoc IsJinkin inverse HyDoc::HasJinks;
    relationship HyNode Isinlinkin inverse HyNode::HasinIinks;
    relationship HyNode Is-outlinkin inverse HyNode::Hasautlinks;
    short create();
    short delete();
    short modify();
    short display();
}

interface clink : HyLink {
    extent clinks;
}

interface ilink : HyLink {
    extent ilinks;
    relationship Set<HyAnchor> Hasilinkends inverse HyAnchor::IsJinkendin;
}
```

![Figure 10: Definition of the HyLink, clink, and ilink classes.](image)

```java
interface HyAnchor {
    extent hyanchors; key anchorid;
    attribute String anchorid;
    relationship clink Isiclinkendin inverse clink::Has-clinkends;
    relationship ilink Iailinkendin inverse ilink::Hasilinkends;
}

interface AnchorIcon : HyAnchor {
    extent anchoricons;
    attribute Bitmap icon;
    attribute short x;
    attribute short y;
}

interface AnchorRegion : HyAnchor {
    extent anchorregions;
    attribute short stmtz;
    attribute short start-y;
    attribute short end-x;
    attribute short end-y;
}

interface Anchor-Node : HyAnchor, HyNode {
    extent anchornodes;
}
```

![Figure 11: Definition of the HyAnchor, AnchorIcon, AnchorRegion, and AnchorNode classes.](image)

```java
interface Paper : HyNode {
    extent Papers;
    attribute String title;
    attribute List<String> authors;
    attribute List<Image> photos;
    attribute List<String> abstract;
    attribute List<Section> sections;
}
```

![Figure 12: Definition of the Paper class.](image)

In current implementation, we use O2 class definition language to specify these schemas. An ODL interface type definition corresponds to an O2 class definition. O2 class definition specifies a structure and a behaviour which is a list of operations attached to this class. O2C programming language is used for the implementation of the operations. Since O2 does not provide the relationship definition, we use the attributes that reference complex objects. For example, the HyNode class is defined as follows in O2.

4 HyOQL query language

4.1 Characteristics

The functionalities of the HyOQL query language result from the two improvements. The first is the set of operations defined in the base classes. It provides functionalities to navigate through hypermedia documents and to query by document contents. The second comes from the extension of the OQL language syntax by defining additional quantifiers.

Querying by content is possible in two ways. First of all, as hyperdocuments are stored in the database, it is possible to directly query the user classes to retrieve documents on relevant user attributes. Querying by textual content is also provided by the operations includes and excludes in the class HyNode.

Structural queries are supported by the operations in the class HyNode and HyDoc. Operations as forwnode and forwlink are available to retrieve the hierarchical structure of the document, i.e., to follow the parent-children and the ancestor-descendant relationships. Operations to retrieve a sub-network of the documents are also provided (e.g. graph). Formulating queries by specifying substructures of a hypertext document is well suited to the characteristic of hypertext as a non-linear information medium.

Hypermedia systems are intended to support applications such as decision analysis for which more powerful quantifiers are needed. In addition to the OQL quantifiers for all and exists, we provide a set of quantifiers such as for many, for most, atleast and exactly that have been introduced by [3]. Introducing this set of quantifiers requires to extend the OQL language syntax. As our query language is based on a SQL-like language, it is more user-friendly than logic based languages.

4.2 Query examples

We now illustrate typical HyOQL queries on the electronic paper reviewing system. As an object query language allows both type properties and operations to be used in query expressions, we can use the operations defined in the base classes anywhere the attributes might be used in a query.
We first show a query concerning the structured documents. For this kind of queries, the OQL path expression is sufficient. The following query finds the third section in the paper identified by P1. \([i] \) is a list expression of OQL that extracts the \((i+1)\)th element of the list collections.

\[
\text{select } p.\text{sections}[2] \\
\text{from } p \text{ in Papers} \\
\text{where } p.\text{id} = "P1"
\]

The next query finds all the papers containing the string "hypertext". It shows a query by textual contents. Includes is an operation that returns true if the node contains the given text string. It is different from the like operator of SQL that is used only to test the existence of a string in an attribute. Includes operation works on all the attributes defined for a node. Notice that the inheritance mechanism is used to call the includes operation. The includes operation is not directly defined in type Paper, instead it is inherited from type HyNode, since Paper is defined as a subtype of HyNode.

\[
\text{select } p \\
\text{from } p \text{ in Papers} \\
\text{where } p.\text{includes}(\text{"hypertext"})
\]

From now on, we address queries concerning the hyperlinking capabilities. The following finds all the outgoing nodes of the paper "P1". It shows a query asking the parent-children relationship. As forwnodes operation is used without role parameter, this query retrieve all the referenced papers and all the notes of the paper P1.

\[
\text{select } p.\text{forwnodes}() \\
\text{from } p \text{ in Papers} \\
\text{where } p.\text{id} = "P1"
\]

The query to find the sub-network containing two nodes connected by a link typed "yes" and whose destination node contains the word "hypertext" may be written in two different ways. The first query starts from the Comment node and acquires all the Paper nodes by applying the forwnodes operation to each Comment node. The second query starts from the Paper node and calls the backnodes operation to get all the incoming nodes, that is Comment node, into this Paper node. Notice that the sub_graph operation is used in the second query to construct a sub_network of paper and comment nodes. As this operation is defined in the HyDoc type, we need to express the path to the HyDoc in order to call this operation.

\[
\text{select } p \\
\text{from } p \text{ in Papers} \\
\text{where atleast } 2 \text{ c in p.backnodes("yes") : true} \\
\text{and not exists 1 in c.backlinksO : true}
\]

\[
\text{select } p.\text{isnode.in.subgraph}(p,c) \\
\text{from } p \text{ in Papers, c in p.backnodes("yes")} \\
\text{where } p.\text{includes}(\text{"hypertext"})
\]

To ask questions about the ancestor-descendant relationship, we need to use the forwpath or backpath operations. The following query finds all the descendant nodes of the paper "P1" by traversing the "ref" links to the level 3.

\[
\text{select } p.\text{forwpath("ref", 3)} \\
\text{from } p \text{ in Papers} \\
\text{where } p.\text{nodeid} = "P1"
\]

Now, we show the use of extended quantifiers. In an electronic paper reviewing system, it is often desirable to ask questions about the papers having positive comments and the comments not having any other arguments. The next query is to find all the Paper nodes having at least two incoming "yes" links and the source nodes of the link "yes" not having any incoming link. The atleast quantifier is introduced. We have also the exactly quantifier.

\[
\text{select } p \\
\text{from } p \text{ in Papers} \\
\text{where atleast } 2 \text{ c in p.backnodes("yes") : true} \\
\text{and not exists 1 in c.backlinksO : true}
\]

The following is to find the papers where there are many arguments "yes" and "no". The for many quantifier is used.

\[
\text{select } p \\
\text{from } p \text{ in Papers} \\
\text{where for many ly in p.backlinks("yes") and} \\
\text{for many ln in p.backlinks("no")}
\]

5 Conclusion
We presented a hypermedia document model together with a complete query language. The model integrates structured and hypermedia documents in the context of the HyTime standard and the object-oriented model. As modeling results, we provide base classes in which all necessary properties and operations are defined. User classes can benefit from these characteristics by inheritance mechanism. The HyOQL language is achieved via these predefined hypermedia operations and a slight extension of OQL syntax. The language is capable of expressing sophisticated hypermedia queries in a very natural way. We comply with
the OQL semantics so that users familiar with SQL-like query languages do not need to learn a new syntax.

Since the class interfaces are defined using ODL, our model can be easily supported by any ODMG-compliant OO DBMS. We are currently implementing our prototype on OO DBMS O2. The O2C programming language is used for the operation implementation. The syntax modifications for the quantifiers can be translated into aggregate functions internally. As for now, we are working on the electronic paper reviewing application without modifying the O2 query evaluator. There are a number of issues that are worthy of further research. These include deeper studies of query optimization for hypermedia applications; specialised graph operators should be particularly considered. We also plan to design a tool box library with functions supporting hypermedia applications.

References


