A Generic Framework for Structured Document Access

Franck Fourel, Philippe Mulhem, and Marie-France Bruandet

Université Joseph Fourier, Laboratoire CLIPS - IMAG
BP 53, 38041 Grenoble Cedex 9 - France

Abstract. Current electronic document retrieval systems, i.e., database systems or information retrieval systems, do not handle enough the richness related to the document structures. Based on a classical model of structured documents, our approach intends to integrate propagation mechanisms related to document structure. These machanisms, called "scopes of attributes", have two goals: they exhibit information that is usually kept implicit in documents, and they address dependencies between the values of the attributes used for the retrieval of the documents. The model of documents, the propagation and retrieval mechanisms compose a generic framework for the retrieval of structured documents.

1 Introduction

Electronic document management is a challenging problem for new information systems, for instance digital libraries. New problems related to the storage and the retrieval of such electronic documents issue from their structural features. Documents are often structured and can be generally viewed as an organization of entities, the document parts. We address here the problem of accessing information from a collection of structured documents. Because of the multiple features of such documents, the retrieval model used has to be adaptable to fit the user's information needs. Most of the time, this retrieval model depends on the "type" of the system: database management system, information retrieval system or hypermedia system, and this model is too strict. We describe here the relevant features of these systems for our context.

Database technology provides sophisticated data management capabilities and query languages. The query results are strict and *A la* SQL queries are structurally oriented; so it is quite impossible to write queries without knowing the structure of the queried data.

Information retrieval systems allow to find a set of relevant information objects, such as documents, corresponding to a user's information need expressed by a query. A "retrievable" document is any information part related to a system internal content representation (called the index). Usual IRSs query languages are closed to natural language, and their matching process compute similarity between the document index and the user's query. Then, IRSs address more flexible retrieval from the content document point of views.

Hypermedia systems manage navigation on any related information parts. With such systems, the retrieval of a document can fail because of the hypermedia size, or by a distraction of the user.

Using the respective advantages of DBMSs and IRSs, we intend then to define a generic framework for efficient and flexible retrieval of structured documents. We address also, at a lower extend, the navigation into the proposed framework.

The framework involves three fundamental elements. We define first which features of documents are interesting for their retrieval (Section2). Then, we express how to declare the mechanisms related to implicit attributes and to the dependencies of the attribute values along the document structures (Section 3). This point is the core of our framework and is dedicated to help the retrieval process of documents that is then defined according to the above elements of the framework (Section 4). An application that implements the framework is then described in Section 5. Finally, the conclusions are given in Section 6.

2 Related works

2.1 Retrieval of structured documents

We discuss here the two connected research topics related to our works: databases and information retrieval on structured documents. Traditional information access in databases are guided by the underlying data structure. In a similar way, structural access corresponds to the location of a specific item in the document structure. For instance, if a user looks for the title and the abstract of scientific papers, he expresses the path in the document structure to reach these items. It is also possible to retrieve items by specifying their related data, their attributes. Such data are for instance the authors of a book chapter or the publication date of a paper and such attribute access is usually provided by DBMSs.

In SGML [Her90], the structure and the attributes are defined in the Document Type Definitions (DTD). So, to formulate a query, a user needs to know some features of the DTD: the attribute names (and their semantic), the types of the document parts described by the attribute and their related position in the whole structure of the documents. To provide these accesses, traditional database query languages have to consider novel operators allowing the definition of structural path. The MULTOS [BR90] and MAESTRO [Mac90] systems offered these operators in the early 90s. More recent work attempts to query structured documents without exact knowledge of their structure using generalized path expressions and path variables [ACC$^+$97].

Access to documents according to their semantic content is related to the retrieval of documents or document parts based on their topic. This retrieval is traditionally provided by IRSs, and requires an indexing process to extract and represent the content of the documents. When managing structured documents to be indexed and retrieved, parts of documents have also to be indexed.

Researches in IR provides results about modeling of structured document content. Passage retrieval, as explained in [Cal94], uses a segmentation of documents that is different from the one provided by the document authors. We consider that the original segmentation of a document by the authors has to be used in such a way to produce sensible results for the user. For that reason, our framework defines the features of the indexing process according to a *semantic structure* of the document directly derived from the syntactic structure of the document.

To define the content modeling, we use studies on the relationships defined in the discourse structure of documents. Two major relationships are exhibited in [Par95]: the *dominance relationship*, and the *intention relationship*. In the discourse structure, each document part corresponds to a set of themes (the themes that appear in the part). The dominance relationships indicates that the composition relationships between document parts are related to the aggregation in the component part of the themes of the composing parts. The intention relationship describes explicit declarations, in the content of one document part, of one or several themes present in another document part. Such intention relationships are usually supported by references between document parts. We want to consider these relations as propagation mechanisms in our works.

2.2 Synthesis

To handle the structural access, our model gives a representation of the documents parts organization, namely the *syntactical structure*. We consider three structural relations: the composition relation defines the aggregation of document parts, the sequence relation expresses the progression of document parts and the reference relation describes the others links between parts (cf figure 1).

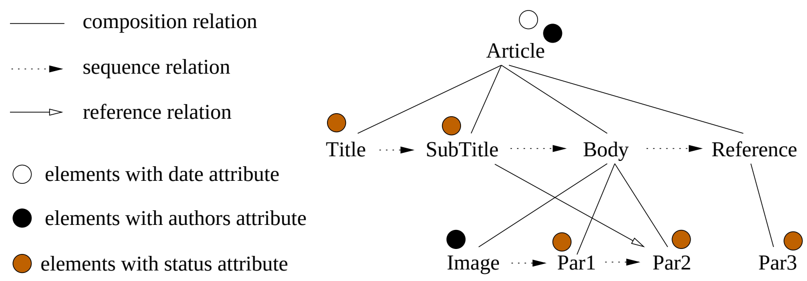

Fig. 1. The attributes of the document D_1

Attribute-based retrieval needs to have integrated attributes within the document model. As we noted above, structural elements admit an attribute if and only if their type allows this attribute. The description is then based on the type of the structural elements. However we will show in the following sections some deficiencies due to the use of such attributes during the querying processing.

We have stated that the data related to a document part is supported by attributes. To be consistent, we also represent the semantic content of document by a specific attribute that depends on the reference and composition structural relationships. These dependencies aim at making explicit the semantic organization of the documents.

Going backwards, can dependencies between attribute values be generalized to other attributes of structured documents ? We show in the following that such dependencies are of great interest and we propose propagation mechanisms on attributes to represent them.

3 Propagation mechanisms in structured documents

The current structured document retrieval systems have some drawbacks due to two essential aspects related to attributes: *coverage* and *dependencies.* We study these and propose to introduce the *scope* of the structured document attributes to address these weaknesses. We show the benefits of the scope of attributes through an example. Then, we describe the whole process that takes into account these scopes : *the structural indexing process.*

3.1 Coverage and dependencies

Some searches on attributes of the initial document D_1 of figure 1 reveal several problems related to the implicit information that occurs in the initial document and the relationships between attribute values.

If we consider that the status attribute defined on the element Par1 is valued by "draft", we shall say intuitively that the elements Body and Article have the same status. While one of their component is a draft, it is sensible to state that they are all also drafts. This information can be deduced from the initial document D_1 but there is no way to represent this in D_1. Consider a query containing a conjunction of criteria on the attributes date, authors and status. We will not find any structural elements of D_1 simultaneously described by the three attributes; so D_1 will not answer the query even if the requested information is present in D_1.

To avoid silence in the answers, we believe that it's essential to make explicit the *implicit information* from the initial document. So we have to propose a way to describe more completely each structural element of document.

In the semantic structure, we have shown which kind of dependencies exist between the semantic description of the structural elements. These dependencies are based on the structural relationships, but they are not defined in the initial document D_1. Here, our aim is to *produce dependencies between attribute values* by the use of the structural relations defined in the syntactic structure: composition, sequence and reference. We want to produce dependencies not only for the semantic content attributes but also for other attributes (date, authors, status, etc).

We are now able to describe more precisely requirements to allow the querying by attributes of structured documents. These requirements are expressed under two essential notions: the coverage and the dependencies of attributes, that correspond to two different processes.

1. We have to create attributes in order to make explicit the implicit informations and to make clear which structural elements are "reachable" (i.e., characterized in any way) by which attributes. That is the *coverage of the attributes*
2. We have then to define which values will be assign to these new attributes. That is the dependencies between the attributes values.

We introduce the scope of attributes in structured documents to tackle these two essential points.

3.2 The scope of attributes

The scopes of attributes in structured documents can be seen as inheritance mechanisms, because they define which structural elements share the same properties. Moreover they make explicit the dependencies between the attribute values.

The *scope of the attribute* α that describes the structural element o of the document D determines the set of structural elements which are concerned by the value of the attribute α on o. The members of this set share the same information source: the attribute α of element o. The scope can be defined by the following tuple :

$$\mathcal{P}^{o}_{\alpha,rel} = (\alpha, o, rel, cat, v_\alpha, f_\alpha, cond)$$

Here, α is an attribute name, o is a structural element, rel is one of the three syntactical relations and cat is a propagation category. We use the syntactical relation and the propagation category to build the set of structural elements concerned by the value of the attribute α of o. If we consider that the predecessors (successors) among rel are candidates to be memberships of the scope of the attribute α, we will have $cat = pred$ (resp. $cat = succ$).

To specify conditions that must be fulfilled by the candidates to the scope of an attribute, we introduce memberships conditions (*cond*. For instance,it's possible to specify that an attribute only concerns some structural types. We specially use these conditions to control which structural elements admit a semantic content attribute.

In the scope, we define not only which elements are concerned by an attribute (coverage), but also the behavior of the values (dependencies): v_α is the initial value of the attribute α and f_α is the propagation function. This last function determines the behavior of the value v_α among the structural relation rel. Using this function, we assign values to the new attributes.

According to the definition of the attributes scope, we have defined four major attribute classes [Fou98]. We will limit our description to two of them : 1) A **static attribute** is a traditional DBMS attribute. Its value describes only the structural element having the attribute ($cat = static$). 2) An **ascending (descending) compositional attribute** (and its value) propagates along the structural composition relationship in a bottom-up way i.e., $rel = comp$ and

$cat = pred$ (resp. in a top-down way, i.e., $rel = comp$ and $cat = succ$). The status and semantic content attributes are ascending compositional, and date and author are descending compositional.

The description above addresses one attribute and one structural element. We focus now on the whole set of scopes on a whole structured document. We introduce the triplet $\mathcal{D}_{\alpha,rel}$ for one document D and one attribute name α: this contains the set OS_α of all the elements of the document D involved in the scopes of α, the set $\mathcal{P}_{\alpha,rel}$ of the scope definitions for α and a relationship rel in D, and finally a combination function, called $combine_\alpha$:

$$\mathcal{D}_{\alpha,rel} = (OS_\alpha, \mathcal{P}_{\alpha,rel}, combine_\alpha)$$

The combination function comes from the facts that i) one structural element has at most one attribute named α, and ii) one structural element may belong to several scopes for α. We must then process a combination of the propagated values from multiple sources, to compute a unique "combinated" value for α. This function is used during the conflict resolution process explained in detail in [Fou98]. The combination function also generates dependencies between the attributes values.

3.3 One example of scope of attribute

Consider the semantic content attribute, called *content*, defined as an ascending compositional attribute on structural type title, subtitle, image, and paragraph, in the document D_1 (figure 1). Its scope for the Image structural element is:

$$\mathcal{P}^{Image}_{content,comp} = (content, Image, comp, pred, Image.content, id_{content}, \emptyset)$$

$Image.content$ represents the original value of the attribute for the Image. The propagation function $id_{content}$ is the identity function, expressing the fact that the elements of the scope receive all the information of $Image.content$. There is no membership condition. This definition states that the Body and the Article elements have a *content* attribute, and the value of their respective *content* attributes depend on the *content* of the Image.

Consider that the others scopes of the *content* attribute are similar for the Title, Subtitle, each Paragraph of D_1. The first step of the process is dedicated to the creation of the *content* attribute on the predecessors of the elements. So, the *content* attribute is created for the Body, Article and Reference elements of D_1 (see figure 2).

The second step assigns values to these new attributes, using the propagation and the combination functions. These functions depend on the attribute domain (the indexing language). We first give the general case where the combination operator is noted $\oplus$ and we show two examples: i) the *content* attribute is a set of keywords, and ii) the attribute is a conceptual graph.

According to the definition of the content scope, we obtain the following relations : Body.content = Image.content $\oplus$ Par1.content $\oplus$ Par2.content, Reference.content = Par3.content and Article.content = Title.content $\oplus$ SubTitle.content $\oplus$ Body.content $\oplus$ Reference.content.

1. If the attribute value is a set of keywords, the combination is then the union set operator. There is then an inclusion dependency relationship between the attribute values, as we show in figure 2.
2. If the attribute value is a conceptual graph, the combination function is then a maximum joint. We used this approach in a medical context with PRIME-GC [MMFB97]. The dependency relationship between the attributes is then a first order logical implication (noted $\phi(a) \rightarrow \phi(b)$).

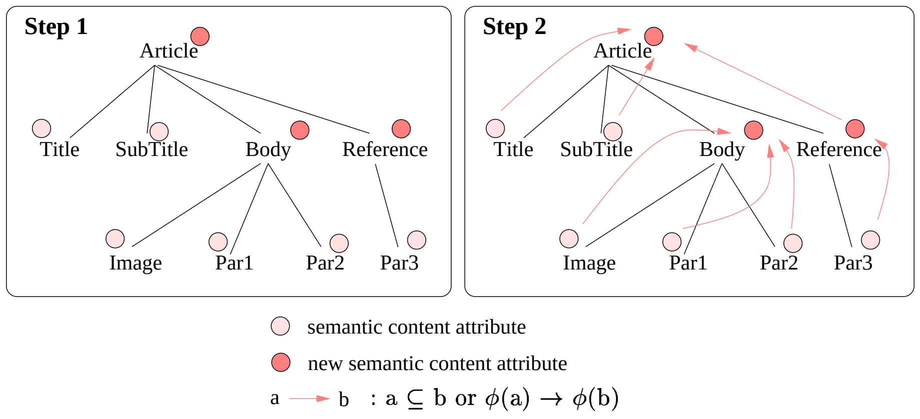

Fig. 2. Scope of semantic content attribute

The use of the scope defines firstly the coverage of the attributes, and secondly computes the attributes values. This last computation uses the propagation function, and possibly a combination function in case of scope conflicts.

3.4 The structural indexing process

The introduction of the scope has some consequences on the document model that we have to manage. In the initial structured document model, we specify the types of the elements that can have given attributes. The use of the scope generates new attributes, and these attributes may not be consistent with the document model. The final document does not now conform any more to the initial document model and so we have to define a final document model to which each final document conforms. The whole structural indexing process is then splitted into two steps :

Step 1: the extension of the initial structured document model, using the attributes scope definitions to be sure that the final documents conform the extended structured document model. The extension is type-related.

Step 2: we apply the scopes of the attributes in the two steps previously described: we create the required attributes and we propagate the attribute values.

The result of this structural indexing process has two majors advantages for the retrieval of structured documents: i) implicit information about the scope of attributes has been made explicit and can then be used during query processing, and ii) the dependencies between attributes can be used to optimize query processing as well as to explain to a user the results obtained by the system.

4 The retrieval strategy

If we consider a traditional query process, we have to test exhaustively and independently each attribute value in the document representation. With our approach, the process takes into account the dependencies and benefits from them. In the following description, we consider the semantic content attribute and the dependency is either the inclusion for keywords, or first order implication for conceptual graphs. The semantic content attributes of the leaves are the minimum attributes, that is the one that have the more precise content.

1. We first test the minimum attributes as is done in a more traditional approach. If a structural element answers the query, all the related elements that include it's semantic content attributes also answer the query. For instance, if the semantic content attribute of the image answers, than the body and the article are also relevant to the query.
2. If the tests on minimum attributes failed, we test the attributes for which all the predecessors (according to the dependency relation) have been tested. For instance, if no minimum attributes of document D_1 have answered, we will first compare to the query, attributes of the Body and the Reference. There is no relevant element if we reach the root of the attributes, In this case the whole article.

Our general retrieval strategy is based on the use of the dependencies created by the structural indexing process. It also depends on the attribute classification. In fact, we develop specific strategies for each attributes classes. The strategy described above concerns an ascending compositional attribute, and starts from the minimum attributes to reach the maximum ones. We define a strategy for each attribute class and each kind of dependency in [Fou98].

5 Application : *My Personal Daily News*

5.1 Overview of the application

We have developed the prototype *My Personal Daily News* devoted to the management of a collection of daily newspapers, on top of the OODBMS O_2. We have chosen an object oriented DBMS because it is argued in [ACC+97] that object database technology is well suited for structured document management. Moreover, we use the O_2Web utility to browse the rough content information stored in any O_2 database with any WWW client.

Our prototype manages newspapers available on Internet and coming from the Liberation server [1]. These documents share the same structure and they involve multimedia data, i.e. texts and still images. We have collected these documents initially as HTML files and we load these into the database as complex objects according to their structure. We store documents in the database as a hierarchy of structural elements and with the raw data attached to the leaves of the hierarchy. Aggregation and list constructors represent syntactic relationships. This approach allows us to keep within the database, the whole structure.

We have automatically indexed the semantic content of the structural leaves of the documents by keywords. For each element, its semantic content is represented by an instance of a subclass of the class ascending compositional attribute. We have defined properties on this subclass to manage the scope of the attribute, that is the propagation function, the memberships conditions and the combination function. So the scope conflicts are handled by these properties.

5.2 Querying and browsing documents

As others have found, we consider that the combination of access by query and by navigation is promising when we deal with a collection of structured documents. In our prototype, we combine the navigation and search processes. Query processing benefits from the results of the structural indexing process.

We have distinguished the queries based on the structural aspects of the document, the queries on the attributes of the document and finally the mixed queries, i.e., queries combining structural aspects and attributes. We deal with *queries by structure* by traditional approaches based on paths access in the syntactical structure. The *queries by attributes* address the attributes that describes the structural elements (classical DB attributes) and also a semantic content attribute. Their computation takes advantages of the generated dependencies between attributes values and we are now able to retrieve elements that classical approaches don't reach. Finally, we allow to combine structure and attributes in queries. The user can specify any kind of criteria and mixes them by conjunction, disjunctions and negations. We handle these queries using sets operators (union, intersection and difference). We have especially focused on the presentation of the query results. We want that the users keep in mind the structural context of these results by showing them the structural position of the relevant items.

We allow users to both browse the whole collection and to complete their access by queries, answers to queries are then considered as starting points in the collection. The *structural navigation* uses the relationships of the syntactical structure to provide hypertext links. On the other hand, the search engine produces hypertext links between the relevant parts. The building of these links is based on the original relationships between document parts. For instance, if two different articles of a newspaper are relevant and even if these ones are not related by the sequence relationships, the matching process relates them to make easier the browsing in the answers.

[1] Libération is a French daily newspaper: http://www.liberation.com

6 Conclusion and Outlook

In this paper, we have defined a generic framework for the retrieval of structured documents that integrates a model of the information flows within structured documents. Within this, new structural elements can be retrieved, and the reasons why documents have been retrieved by the system are clearer for a user.

The retrieval of the good "information" is becoming a salient point with the current communication facilities, we will then focus in the future on the following points: the kind of answers given to the user, and a better integration of the user.

In some contexts, like electronic newspapers for instance, a same information can be presented under different points of views with different newspapers. It is then interesting to give as an answer documents built-up by relevant parts of documents. The same approach can be used to generate relevant documents if not any exists in the document base.

On the other hand, we intend to bring back the user into the core of the retrieval process. We consider that indications provided by a user about the relevance of the retrieved documents should allow an evaluation of the information flow within structured documents. Such results could improve the overall system quality by tailoring the scopes in a way to satisfy the user.

Acknowledgements: We would like to thank Pr. Alan Smeaton for his help in polishing the english version of this paper.

References

[ACC+97] Serge Abiteboul, Sophie Cluet, Vassilis Christophides, Tova Milo, Guido Moerkotte, and Jerom Simeon. Querying structured documents in object databases. *International Journal on Digital Libraries*, 1(1):5–19, April 1997.

[BR90] Elisa Bertino and Fausto Rabitti. *The MULTOS Query Language*, chapter 4, pages 53–74. North-Holland, 1990.

[Cal94] James P. Callan. Passage-level evidence in document retrieval. In W.B. Croft and C.J. van Rijsbergen, editors, *Proceedings of ACM SIGIR'94*, pages 302–310, Dublin, Ireland, July 1994. ACM.

[Fou98] Franck Fourel. *Modélisation, indexation et recherche de documents structurés.* PhD thesis, Université Joseph Fourier, February 1998.

[Her90] Eric Van Herwijnen. *Practical SGML.* Kluwer, 1990.

[Mac90] I.A. Macleod. Storage and retrieval of structured documents. *Information Processing & Management*, 26(2):197–208, 1990.

[MMFB97] Mourad Mechkour, Philippe Mulhem, Franck Fourel, and Catherine Berrut. PRIME-GC: a Medical Information Retrieval Prototype on the WEB. In *RIDE'97. Seventh International Workshop on Research Issues in Data Engineering*, Birmingham, England, April 7–8 1997.

[Par95] François Paradis. Using linguistic and discourse structures to derive topics. In *Fourth International Conference on Information and Knowledge Management (CIKM)*, pages 44–49, Baltimore, Maryland, USA, November 1995.