2000

# Structuration and Enrichment of HTML Documents in order to Build a Specific Information Warehouse

Josiane Mothe
*IRIT/Equipe SIG*, mothe@irit.fr

Franck Ravat
*IRIT/Equipe SIG*, ravat@irti.fr

Farshad Riahi
*IRIT/Equipe SIG*, riahi@irit.fr

Gilles Zurfluh
*IRIT/Equipe SIG*, zurfluh@irit.fr

# Structuration and Enrichment of HTML Documents in order to Build a Specific Information Warehouse

**Josiane MOTHE, Franck RAVAT, Farshad RIAHI, Gilles ZURFLUH**
I.R.I.T. / Equipe SIG
118 route de Narbonne
31062 Toulouse Cedex 04, France
{mothe | ravat | riahi | zurfluh}@irit.fr

***Abstract* - This paper presents a process to enrich the web document representation in order to supply an information warehouse and allow more precise queries than the web search engines do. This information warehouse is stored in an object-oriented database (OODB) so that powerful set-based query languages can be used. One of the main contributions of the paper is the HTML document enrichment while supplying the warehouse. This enrichment is based on the document decomposition and on the components indexing. These processes take into account the logical and the hyperlinking structures as well as the appearance of the Web documents. A prototype has been developed using the OODBMS O2.**

***Keywords* - Web page indexing, logical structure extraction, hyperlink structure extraction, data enrichment.**

## I. INTRODUCTION

The most popular way to retrieve information related to a specific subject on the web, is to query (meta-) search engines. These tools generally retrieve long lists of matching URLs. The matching process is based on the fact that the keywords the user puts in his/her query occur somewhere in the retrieved URL contents, whatever the web page length. The user has then to select the URL and to skim through the all page to find the possible relevant information chunks. Passage retrieval functionalities are not provided by web search engines despite the fact that they have been shown to be efficient in structured document context e.g. using SGML documents [23], [9].

Web search engines do not either implement operators permitting to specify mixed queries, on the structure of the web pages and on their content, at the most some operators allow one to query some logical structure elements (title, URL, …). However, in the literature, some works report query languages that combine structure-based and content-based queries [7], [11]. Most of these works propose to store the structured documents in databases so that a set-oriented query language can be applied to the data and this solution seems to be promising.

The techniques that take advantages of the document logical structure have been shown to be efficient in the context of well-structured documents [1], [7], [23]. They improve the querying possibilities and the relevance of the retrieved elements. Unfortunately these techniques cannot be applied directly to non well-structured documents such as the web documents. One of the reasons is that the mark up language used until now on the web (HTML), is generally used by the document authors to physically structure the document (for document appearance) rather than to logically structure it (marking up its composition).

The main issue of our work is to deeply turn to account the web document structures in order to provide a rich representation of these documents and thus to be able to answer more specific user needs. The user can use mixed queries based on the document structures and on the content it-self. S/he can also choose the granularity of the retrieved information chunk. More precisely, this paper presents a solution to automatically build an information warehouse or dataweb from one or several web servers or from a set of web pages on a given domain. Indeed, the enrichment of the information is not accurate for the all web as it is space and time consuming but rather to a specific part of it from which the users can find specific information. While supplying the dataweb, the documents are pre-treated and enriched by adding different data and meta-data that allow a more efficient access to the needed information. More precisely, a descriptor is associated with each chunk of information. This descriptor is composed of terms representatives of its contents as well as some more specific meta information extracted using information extraction techniques. The chunk detection is based on the web document structure (the web page logical structure).

This paper is organised as follows. Section II presents the related work. Section III describes how the web documents are modelled to be stored into an OODB. Section IV details the processes used to deduce the document structure from the HTML tagging. Section V describes the indexing task. In section VI, the global system architecture is presented. Conclusions and perspectives are discussed in the last section.

## II. RELATED WORK

Different works have proposed solutions to store structured documents in a database. A DBMS provides relevant structures to store documents and a set-oriented language to query them. In that case, a document database is composed of two elements: text (sequence of symbols) and structure (set of independent hierarchies) [16], [10].

With regard to SGML documents [12], the logical structure is marked up using tags defined in a DTD (Document Type Definition). As a result their logical components can be easily extracted to be stored in an object-oriented database (OODB) [7] [1]. OODB provide all the needed features (sophisticated structure type, query language that can easily be extended). [7] highlights the extraction of the logical structure of SGML documents for the implementation of an information system based on a DBMS. Storing SGML documents in a database permits to retrieve text, by content-based or by structure-based queries [25]. These approaches could be easily generalised to web documents written using the XML language [15]. However most of the documents on the web are written using the HTML language [5], an extension of the SGML language. Compared to SGML, HTML is more

especially used to display a document on a screen than to mark up its logical structure.

However, [11] defines a meta-data extraction mechanism that gives a structure to the HTML documents. The structuring process is based on several regular grammars (one per document type: scientific papers, technical reports, …) that guide the interpretation (itself based on meaningful tags. The object-oriented model they propose integrates the logical document structure, the hyperlink structure and some information about servers. The problem of this solution is that the writer has to structure the HTML documents according to the rules of a specific regular grammar. Thus, this solution is adapted for an Intranet (or for a type of documents as research papers) but not for the Internet where the documents are very different in their contents and their appearances. The WIND model gathers information on a given topic from different web sources and organises them in data repositories [9]. This integration is based on several grammars stored within a library. Each of them is used to describe a specific kind of HTML documents (personal homepages for example).

NoDoSE [2] is a system that infers the structure of a collection of similar documents by combining automatic analysis with user's input. Before extracting data from the documents, the users have to define the structure of the documents. The advantage of this solution is that this system is able to infer plain text documents or HTML documents.

## III. DATAWEB STRUCTURE

The dataweb corresponds to an information warehouse build from web pages. It does not solely store the web pages URLs and the indexes that permit to select the URLs that match the query terms, as web search engines do. In addition, the dataweb memorises the raw information, reorganised and enriched it.

The dataweb structure is inspired from models used when modelling well-structured documents [10], [8]. The model has to be generic enough to be applied to any web pages and complete enough to allow queries based on document content and/or document structure. In that way, richer query facilities than the ones provided by web search engines will be provided.

The dataweb model takes into account different aspects of the web pages: their logical structure, hyperlinking structure and their content itself.

### *The logical structure*

With regard to the logical structure, a web page is considered as composed of parts that can be in turn split up into sub-parts and so on. It can be represented as a tree where the nodes correspond to abstract entities, named Informational Objects (IOs), and where leaves correspond to concrete entities, named Informational Units (IUs).

Our dataweb model is based on the object paradigm in order to model easily multimedia and complex data occurring in web pages. So, the IO and IU entities are modeled with two classes (see Figure 1).

Each IO has a title and a level in the web page decomposition hierarchy. Other meta information is associated with the *IO* class, including the IO *length* (e.g. significant word number), the *confidence degree* and the l*abel*. The length attribute is useful both for the indexing process and for querying IOs ("*Retrieve IOs dealing with OODBMS that have less than 200 words*"). The confidence degree attribute can be defined according to the incoming link number [6]. The more this number is big, the more the degree of confidence is important. This degree can be used during the retrieval process in order to rank the retrieved IOs. The label attribute is instanciated with "labelX" when a <A Name="labelX"> tag occurs just before the IO, otherwise this attribute is null.

Each web page is an IO which is characterised by its URL name and eventually by some meta information (localised in the head part of the page, like authors names, last update) that refers to a specific web page.

IUs are elements that cannot be broken down without losing an important part of their semantics. Paragraphs, images, sounds, lists and tables are considered as IUs. The representation of web pages according to this structure allows component retrieval and queries on structure. As an example, it is possible to retrieve all the images that are included in some selected web pages or to retrieve the documents that have a section containing a given phrase in its title. All these elements are modelled in the *Informational Object*, *Web page*, and *Informational Unit* classes and ad-hoc classes that inherit from the *IU* class. Figure 1 represents our dataweb model depicted with the UML [22] formalism.

### *The hyperlinking structure*

The **hyperlinking structure** considers

- the outgoing links. They correspond to links from the current IO to IOs from other web pages or from the same page,
- the incoming links. They correspond to links from other IOs to the current IO,

The anchor of a hyperlink is considered as an IU whereas the destination of the link is an IO (either a whole document or a document part when a <A NAME> tag is used).The hyperlinking structure is modelled in the dataweb model (see Figure 1) either by an association between the *IO* and *IU classes* when the corresponding objects are stored in the dataweb or by an attribute of the *IU* class (outgoing links) when they are not stored in the dataweb.

The storage of the hyperlink structure permits to model the semantic relationships defined by the authors between web page components. Thus this model fits more with the graph structure of the Web. This hyperlink storage also provides a complementary way to access either web pages or their components. For instance, all the IOs referenced by a given IO can easily be found, as well as all the IOs which refer to it. In addition, this hyperlink structure will be used to enrich component indexing (see section V).

### *The page content*

The web pages are analyzed according to a third aspect which is the page content. This analysis is based on a specific indexing process (extracting a set of representative terms and associating a weight to each indexing term). Our indexing process integrates the three following analyses: the term

frequency in a IO, the term appearance and the hyperlinked IOs.

In our model, we represent each IO by a set of indexing terms. This representation is modelled as an association between the *IO* and the *indexing term* classes. Each association has a property which is the term weight in the IO. This weight is defined by the term frequency and appearance analyses.

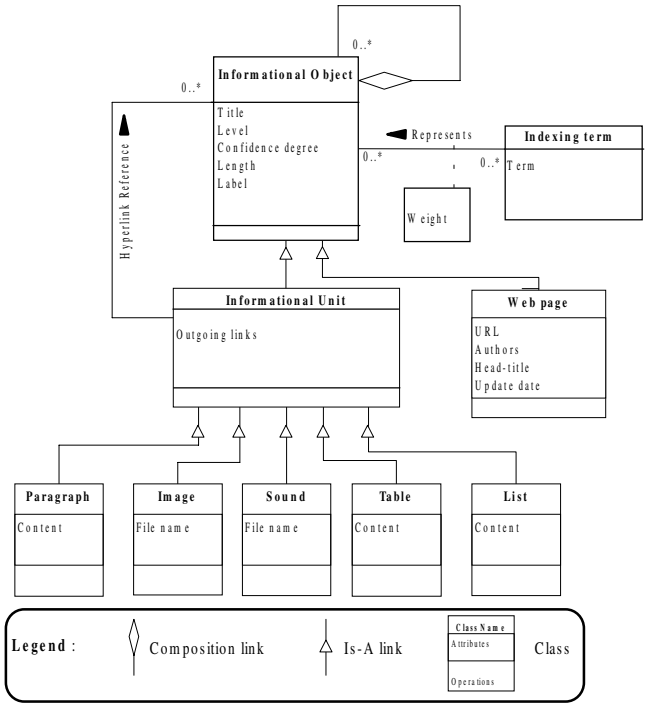



**Figure 1** : The dataweb meta-model.

The last aspect of the web page analysis corresponds to the possible adaptation of the generic model to a more specific one. The analysis of some specific pages reveals that it can be relevant to add, for example, some attributes in the *Web page* class which correspond to meta-information such as the page type, the page authors and so on.

In this section, we defined our dataweb model. It integrates the logical structure, the navigational structure of the HTML pages and the content it-self. In the following sections, we focus on the extraction of the HTML document structures and on the content analysis (indexing process).

## IV. EXTRACTION OF THE DOCUMENT STRUCTURES

The goal of this section is to describe the process used to extract the structures (logical and hyperlink structures) of HTML pages and to create objects in the appropriate classes of the dataweb schema we defined (instantiation process).

Extracting automatically the logical and the navigational structures (without human operator) from web pages implies an analysis of the HTML tags. This analysis permits to instantiate the object classes. We distinguish several categories of HTML tags [18] as follows:

*(i)* presentation tags (e.g. <B>, <I>, <BLINK> ),

*(ii)* structural tags[1] (e.g. <H*i*> … </H*i*> where i=1..6, <P>, <TABLE>, <UL>),

*(iii)* inlaying tags (<IMG SRC="….">, <BGSOUND SRC="….">),

*(iv)* referring tags (<A HREF="….">),

*(v)* informational and meta tags (e.g. <META NAME="Authors" CONTENT="…">, <META NAME="keywords" CONTENT="…">, <ADDRESS>).

### A. *Extraction of the logical structure*

Our goal is to decompose a web page into its logical entities according to the dataweb model we defined. So, we analyse the inlaying and the structural tags of each HTML document.

In contrast with the SGML or XML languages, the HTML tags do not specify the beginning and the end of the structural elements. In addition, web page authors still mostly use the HTML tags for appearance purposes (document physical structure). Thus, one of the challenges is to add more semantics to the tags so that the logical structure of the web pages can be deduced. In fact, with HTML pages, the recognition of the logical structure beginning is based on typographical information attributes e.g. enlargement of titles using different font sizes according to the section level, line feed to define a paragraph, … For instance, when the reader sees a new title, s/he deduces that the previous paragraph and the previous section are ended.

We provide an automatic process that analyses tags used in a web page and instantiates the dataweb classes.

#### *a. Instanciation of the Web page class*

The analysis of an HTML file first leads to the creation of a *Web page* class instance. This instance is identified by the *URL* attribute. The analysis of the meta tags from the web page head permits to instanciate additional meta-information attributes such as the *authors*, *date*, …. attributes.

#### *b. Instanciation of the Informational Unit class*

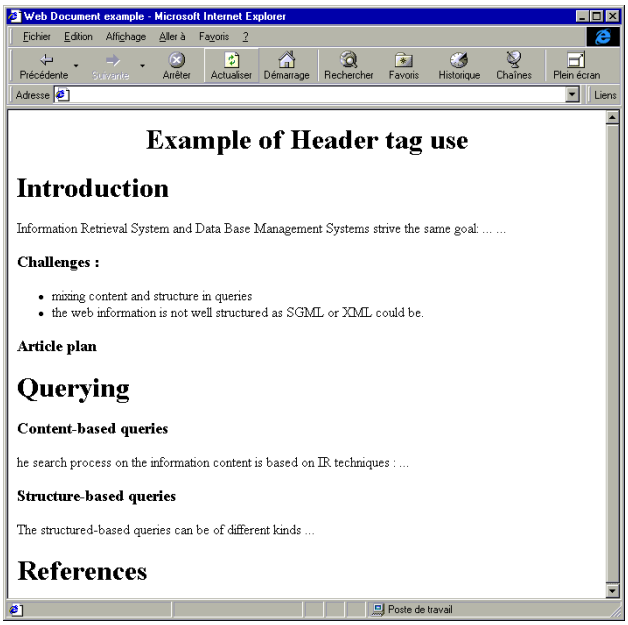
The creation of IU instances is based on the recognition of the inlaying tags (<IMG SRC=... >, <BGSOUND SRC=... >), and of some structural tags (such as <P>, <UL>, <OL>, <TABLE>). Each occurrence of one of these tags instanciates a new IU in the database. The data in between these tags and the corresponding closing tags (</P>, </UL>...) is used to give a value to the *content* or *file name* attributes. In fact, a deeper analysis is done to detect weak use of tags and missing closing tags. This latter analysis takes the HTML DTD into account.

#### *c. Instanciation of the Informational Object class*

The IO class instantiation consists in detecting the components of a web page (in addition to the IU and web page IO). This task is difficult as there is no specific HTML tags to mark up the components and their levels in the document hierarchy. Indeed, the HTML DTD provides <H*i*> tags that are generally used by the authors to highlight the

1 Often these tags are used for appearance purposes (to highlight some titles, include some blank space,.. ) but do not mark up the document logical structure in a relevant way.

section titles. Unfortunately, this DTD is not strict enough so that some authors use these header tags in a wrong way. As example, the authors of web pages can shift the use of the header tags : thus they can use the tag <H2> for sections of level 1 and the tag <H4> to mark up sections of level 2 (see Figure 2). In fact, in this case, they attach more importance on the appearance that the page will have when displayed than to the logical structure marking up.




```
<HTML>          <HEAD>
<TITLE>Web Document example</TITLE>
</HEAD>
<BODY>
<H1 align=center>Example of Header tag use</H1>
<H1>Introduction</H1>
<P>Information Retrieval System and Data Base Management
Systems strive the same goal: ...</P>
<H3>Challenges :</H3>
<UL>
<LI>mixing content and structure in queries</LI>
<LI>the web information is not well structured as SGML or XML
could be. </LI>
</UL>
<H3>Article plan</H3>
<H1>Querying</H1>
<H3>Content-based queries </H3>
<P>he search process on the information content is based on IR
techniques : ...</P>
<H3>Structure-based queries</H3>
<P>The structured-based queries can be of different kinds ...</P>
<H1>References</H1>
</BODY>
</HTML>
```

In this example, the author uses a <H3> tag to mark up a title of level 2. Our algorithm is used to clean this kind of header tag weak use.

**Figure 2**: Example of a HTML page (source and appearance).

Even if the beginning of a section (IO) can be detected when a <H*i*> tag occurs, the difficulty remains in determining the level and the end of this section (its range). To solve this problem, we define the following rules:

- If a *header* tag <H*i*> (i from 1 to 6) appears only once within the document, it is assumed that it is not marking up a structural element (IO),
- When the *header* tags appear more than once, we consider that <H*x*> marks up a title of level y, if and only if:
  - for y=1, $\forall$ i $\in$ [1..x-1], <H*i*> does not exist.
  - for y > 1, $\exists$ z $\in$ [1..x-1], <H*z*> marks up a title of level y-1, and $\forall$ t $\in$ [z+1..x-1], <H*t*> does not exist. By convention, if the lower limit of an interval is larger than its upper limit, it will be regarded as empty.

Using these rules, we detect the beginning and the level of the sections. The range of a section detected is defined as follows: it begins with the <H*i*> tag and ends just before a new occurrence of <H*i*> or at the end of the document. A section is modelled as an instance of the *Informational Object* class.

**PARTICULAR CASE**: the <H*i*> (i from 1 to 6) tag when occurring at the very beginning of the document may correspond to the page title or to a section title. It is considered as the page title if it is directly followed by an other <H*i*> of the same level or if it is the only <H*i*> of that level in the document. It is considered as a section title otherwise.

To implement this process, we use a rewriting technique. It allows us to determine the beginning and the end of the IOs. Below, we briefly explain the principle of rewriting we use.

### *d. Document re-tagging (rewriting)*

This process rewrites a HTML page in a pivot language that specifies the beginning and the end of each IO (kind of HTML2XML tool). This process provides a well-formed document according to the XML description. We use the ExRep tool [Lambolez et al. 95], developed in our research team, to achieve this document re-tagging. This tool utilises the regular expression matching to recognise character sequences or words built with an alphabet. If a word matches a regular expression within a context, the tool applies a process or an action to it e.g. replacing matching word by another word, rewriting it, updating a counter or activating a new context. The combination of such regular expressions with the actions to do defines a rule written as follows:

***expression to recognize=expression to rewrite;action to do ;***

In ExRep, a set of such rules constitutes a filter. We use ExRep tool with several filters applied sequentially to rewrite HTML tags as predefined tags (see Figure 3), applying the criteria of the IO decomposition given above. After this stage, we obtain the beginning and end of the IOs. Then the creation of the instances of the object classes is possible.

***Extraction of the hyperlink structure***

The extraction of the hyperlink structure is based on the analysis of the <A HREF> tags used by the web page author.

Generally, the anchor of hyperlink is a phrase or an image whereas the destination can be either an URL (e.g. http://www.irit.fr/welcome.html) or a label in an URL (e.g. http://www.irit.fr/welcome.html#fromAirport).

```
@ Main Context
|\<META NAME\="(keywords|KEYWORDS|Keywords)"
.....\="<.+>"\>=\n Motscles : \0\n;;
|\<H1 (align|ALIGN)\=(center|CENTER)\>=\n document
title\n;PushContext(recup_title), SetVar(level,1);
|\<\<H1\><.+>\</H1\>>=\n\<Level1Title\>\n\2\n\</ Level1Title \>\n;;
|\<\<H5\><.+>\</H5\>>=\n\< Level5Title \>\n\2\n\</ Level5Title \>\n;;
|\<(i|I)\>=\n Keyword : ;PushContext(recup_Keyword);
|\<(b|B)\>=\n Keyword: ;PushContext(recup_Keyword);
|\<(strong|STRONG)\>=\n Keyword: ;PushContext(recup_Keyword);

@ recup_title
|\</TITLE\>=\n\n;PopContext();
|[\x00-\xFF]=\0;;

@ recup_Keyword
|\</(i|I)\>=;PopContext();
|\</(b|B)\>=;PopContext();
|\</(STRONG|strong)\>=;PopContext();
……
```

*The above ExRep rules allow to extract the document title, sections titles, keywords highlighted by typographical changes, etc. The rules are associated to a context (defined by @ context_name) and can be activated only if the context is active. The context can be encapsulated and are activated using PushContext() and deactivated using PopContext() functions.*

**Figure 3**: Example of rewriting rules applied to a HTML page.

In the dataweb, the anchor of a hyperlink is considered as an IU (containing the <A HREF> tag) whereas the destination of the link is an IO. In our model, the hyperlink destination IO is either a web page or an IO having a label (object of the IO class with the *label* attribute valued during the logical structure specification).

The hyperlink structure extraction consists:

- in updating the IU *outgoing* attribute if the pointed IO doesn't exist in the dataweb, or
- in instanciating an association link between the anchor IU and the pointed IO (a web page identified by the HREF tag URL or the component of this web page having the same label than the HREF label).

The algorithm Figure 4 describes this process.

This enrichment of the dataweb information based on the web page structure analysis (logical and hyperlinking) is completed by a content based enrichment.

## V. IO INDEXING

Instead of indexing the web pages in their whole, the indexing process is applied to page components. It has been shown to be efficient when one want to do passage retrieval (see [23] as an example in the SGML context). In our approach, the indexing corresponds to the third level of the dataweb information enrichment (the content analysis). As explained in the section III, a web page is composed of IOs. The IOs of the lowest level (IUs) contain the concrete information. Thus, we chose to first apply the indexing to the IUs.

```
HyperlinkDefinition(CurrentIU,HREF_ADDR)
{We assume that the HRED_ADDR is defined as
follows: HREF_URL#HREF_LABEL}
Begin
If not exist a Web Page with WebPage.URL =
HREF_URL
   Then
    CurrentIU.outgoinglinks:=HREF_ADDR;
   Else {the web page identified by HREF_URL
exists}
     If HREF_LABEL=" "
     Then
     instanciate an association link between CurrentIU and
the IO representing the webPage.
     Else
      If not exist an IO component of the
WebPage with IO.label=HREF_LABEL
      Then instanciate an association link
between CurrentIU and the IO representing
the webPage.
      Else instanciate an association link
between CurrentIU and the IO
End
```

**Figure 4**:. Component detection algorithm.

In fact, to allow a user to retrieve any IOs, the retrieval process has to compare the user's query with the IO representations (not solely with the IU representations). The IO representations can be computed during the matching process or during the indexing process, according to the IU representations resulting from the indexing. We choose the latter case to minimise the request response time. This indexing process involves widely used methods [19] that we complete by techniques that take into account the web specificities.

### *Recursive indexing process*

To index the IOs, we apply a recursive process. The first IOs that have to be indexed are the IUs. Thereafter, a bottom-up return is used to determine the indexing terms which represent an IO of higher level. The set of indexing terms associated with an IO is defined by the union of those associated with its components. At each level, the term indexing weight is re-computed in order to define a more accurate IO representation (according to the formulas given in the next section). To limit the indexing term number, a threshold can be defined so that an indexing term will be kept only if its weight is higher than the threshold.

### *Indexing techniques*

We use several techniques to determine the indexing terms of an IO as well as their weights. These techniques are based respectively on the following analyses:

- the frequency of the terms contained in the IO ,
- the presentation of the terms contained in the IO,
- the hyperlinked IOs

The first technique takes up classical methods, whereas the two others are based on the Web specifities.

*a. Indexing term detection*

An indexing term can be either a single word or a phrase. The detection of phrases is important in order to represent the IOs by word groups or phrases in addition to single terms.

To determine the single words to be retained, we use a stop list (to remove useless terms) and the Porter's stemming process [17]. Concerning the phrases, they are defined as a sequence of terms that occurs frequently [19]. In addition, the analysis of the component term appearance is used to choose the phrases that will be finally attached to an IO. More precisely, when the terms from a detected phrase had not been assigned the same appearance (normal, bold, italic, …) we assume that the author does not mark up the term sequence as a phrase, and we do not consider this phrase as an indexing term for the current IO.

Once detected, the indexing terms are weighted. These weights are computed using classical term frequency based formulas and novel techniques based on term appearance and semantic links between IOs. The final indexing term weights are computed combining partial weights that result from each technique.

*b. Term frequency partial weight*

The first partial weight is computed according to the fact that the characterisation power of a term is proportional to its frequency and to its inverted document frequency. In [20], a logarithmic formula was proposed to calculate the term weight in a document. [21] presents a N-based logarithmic weighting function (where N is the term number). We adapted these formulas in order to obtain the indexing term weight for an IO.

$$\text{Weight}_{\text{freq}}\, j,i = \frac{\text{TF}_{j,i}}{\text{Long}_{i}} * \text{Log}_{\text{NIO}}\,(\text{N}_{\text{IO}} / \text{N}_{\text{IOj}})$$

Where $\text{TF}_{j,i}$ is the term frequency of term j in the IOi, $\text{Long}_{i}$ is the IOi length, in regard with significant word number, $\text{N}_{\text{IO}}$ is the current total number of IOs, and $\text{N}_{\text{IOj}}$ is the current total number of IOs containing the term j.

This formula takes into account the number of IOs. Because the IOs are indexed when added in the dataweb, the weight formula takes into account the current number of IOs (this number evolves when adding new web pages). An N-based logarithmic weighting function [21], [18] has been shown to be more stable when new documents are added.

*c. Term appearance partial weight*

To complete the term weighting, we use a complementary mean which takes use of the font parameters the authors chose in order to highlight some phrases. Indeed, the presentation tags of HTML language make it possible to highlight text, by changing its format or its typography (character boldface, italic, size or color). These typography changes are implemented by tags, such as <I>, <B>, <Strong>, <Font color="#rrggbb">, etc. The authors of the web documents usually use this change of typographical style to highlight a significant concept. Thus, the analysis of such tags makes it possible to determine a significant concept.

To calculate the partial weight of the term highlighted with these tags, we proceed as follows:

The weight of a term *j* in the IO *i* is proportional to the ratio of the number of times it is highlighted using the presentation tags ($\text{N}_{\text{tag}}$) by its frequency in the IO ($\text{TF}_{j,i}$). Hence, the formula is as follows:

$$\text{Weight}_{\text{tag } j,i} = \text{N}_{\text{tag}} / \text{TF}_{j,i}$$

This partial weight is included in the interval [0..1].

*d. Partial weight with regard to the hyperlinked IOs*

A partial weight is computed according to the IOs that are linked to the current indexed IO. When an author decides to refer an URL in a web page, s/he expresses a semantic link between the linked contents. We use this semantic link in order to complete the IO indexing. As said before, in our model, an hyperlink is defined between an IU and an IO. Hence, while indexing the IO corresponding to the IU that is the hyperlink anchor, we consider the indexing terms of the destination IO if already indexed (je ne comprends plus ce que l'on a voulu dire…). The formula used to compute this partial weight for the term j in the $\text{IO}_i$ is the following one:

$$\text{Weight}_{\text{hyp ji}} = \text{Weight}_{\text{jk}}$$

Where $\text{Weight}_{\text{jk}}$ the final weight of the term j, which is the hyperlink origin of $\text{OI}_k$ pointing to $\text{OI}_i$.

*e. Final indexing term weight*

Once the partial weights of each term have been computed, (by using the frequency, appearance and hyperlinks) the final weight of an indexing term *j* in the IO*i* ($\text{Weight}_{\text{ji}}$) is computed as follow:

$$\text{Weight}_{\text{ji}} = ((\text{Weight}_{\text{Freq ji}} * \text{C}_{\text{Freq}}) + (\text{Weight}_{\text{tag ji}} * \text{C}_{\text{tag}}) + (\text{Weight}_{\text{hyp ji}} * \text{C}_{\text{hyp}}))/ (\text{C}_{\text{Freq}} + \text{C}_{\text{tag}} + \text{C}_{\text{hyp}})$$

The $\text{C}_{\text{Freq}}$ $\text{C}_{\text{tag}}$ and $\text{C}_{\text{hyp}}$ coefficients are chosen according to the importance one wants to give to each of the features (term repetition, term highlighting, hyperlink). Assessment will be needed to tune the most efficient values (as a starting point, one can choose the same value for each coefficient).

***Index storing***

To each IO instance is associated a set of *Indexing term* objects. Each association instance between an IO and an indexing term is defined by a weight (see Figure 2). This association allows:

- to find the IOs characterised by a single word or a phrase given by the user,
- to obtain the set of the terms which characterizes a given IO,

Thus, this association makes it possible to retrieve some document components, starting from the terms chosen by the user.

## VI. System Architecture

The tool, we developed automatically instanciates a dataweb starting from web pages. Figure 5 presents the system architecture. The Parse Engine is used to specify the logical structure. The Information Extracting Engine permits to define the navigational structure. Finally, the Indexing Engine associates a set of indexing terms to each IO.

Instead of querying the web through a web search engine, we offer the users to query directly a warehouse which contains the information gathered from several chosen sites and enriched. This information is organised according to different dimensions that allow more complete queries that those done using web search engines.

Our tool is implemented using the O2 system in O2C (about 7000 lines of code). The resulting objects from the web page enrichment treatment can be displayed using the O2 tools (Figure 6 & 7). Navigational facilities are offered to the user:

- to browse through the logical document structure from the retrieved IOs,
- to browse through the hyperlinking structure.

As the dataweb is stored using the OODBMS O2, all the querying facilities provided by this system (OQL, O2Web) can be used. In addition, the query language could be extended to allow the combination of structural and content-based queries as done for SGML and HTML documents [7], [11], [4].

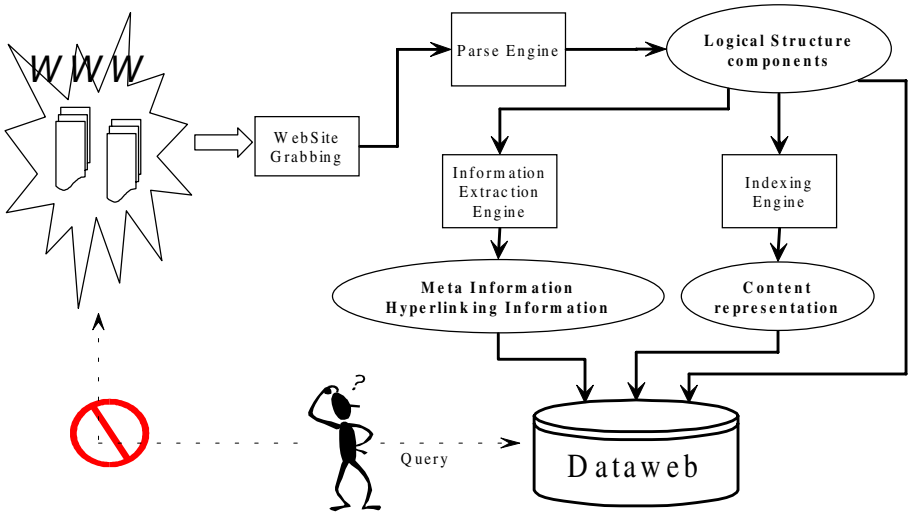



**Figure 5** : System architecture

## VII. Conclusion

To retrieve information on the Web, the web search engines provide a query language to retrieve documents based on the whole document content. These tools are not satisfactory when searching a specific information on a subject. In this paper, we present a solution to supply an information warehouse build in advance from web pages or web servers and to improve the querying possibilities and the relevance of the retrieved information. This dataweb deals with HTML documents. We based our work on the HTML language as it is the most commonly used language to write web documents but the approach used can be easily extended to other description languages such as XML documents (indeed we use this language as a pivot format in our model).

This dataweb stores web pages after having enriched them. This enrichment considers several aspects of a web page: the logical and the hyperlink structures as well as the web page appearance and the content itself. We automatically extract the logical structure and define more accurate chunks of information (IO) so that passage retrieval is possible. These IOs are defined by analysing the HTML tags and the web document logical structure is induced from that analysis. By doing this, we then provide information in a format that allows queries based on the page structure. With regard to the hyperlink structure, we integrate the outgoing and the incoming links of the IOs. The integration of these concepts within the dataweb is based on the exploitation of specific HTML tags. Finally we characterise each IO with their representative terms. We define a specific indexing process based on complementary techniques which are the analysis of the term frequency, the appearance of the terms contained in the IOs and also the hyperlinking analysis.

The presented solution has the following advantages:

- the process is based on a generic model of web pages, it is not limited to a specific context like the methods based on regular grammars,
- the chosen data structure can be specialised, for example, adding new attributes corresponding to meta information,
- the model integrates the point of view of the web page writer. This point of view is transcribed through the

appearance s/he gave by highlighting some phrases and through the hyperlink definitions,

- the using of an OODBMS to store the dataweb is efficient for hypermedia information and provides powerful set-based query languages. This kind of languages can be extended in order to allow mixed queries based on both structures and content information. These functionalities would allow the specification of complex queries that web engines cannot currently offer (for example, the list of the authors referenced in all the documents on "information retrieval").

We implemented the proposed dataweb on O2. The tool that extracts the logical and the navigational structures as well as the HTML document content is implemented in O2C on Sun Workstations and uses the ExRep tool. The next step of the project is to implement the extension of the OQL and WebOQL and to provide the users with a more accurate interface to query the dataweb.

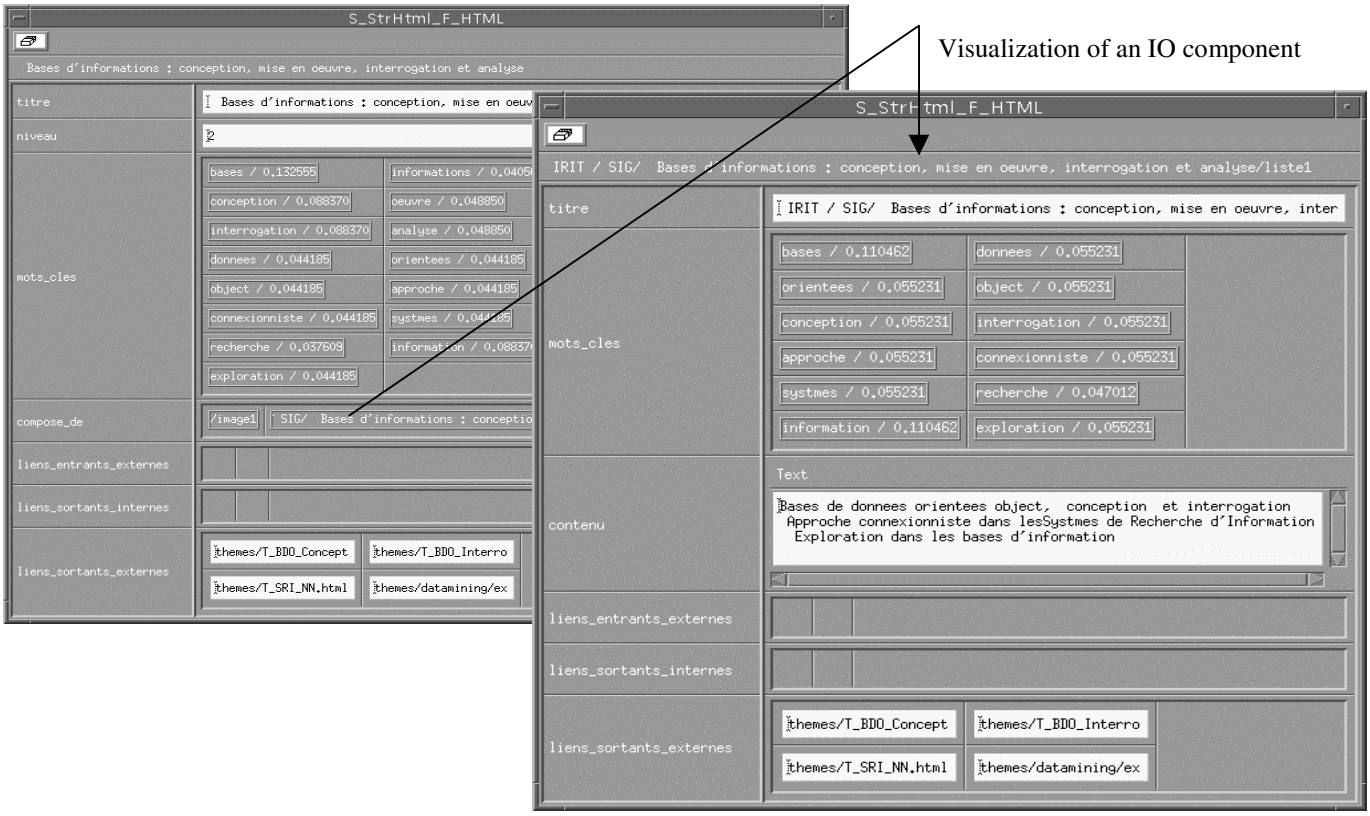



**Figure 6** : Dataweb IO visualisation and navigation (Using O2)

search

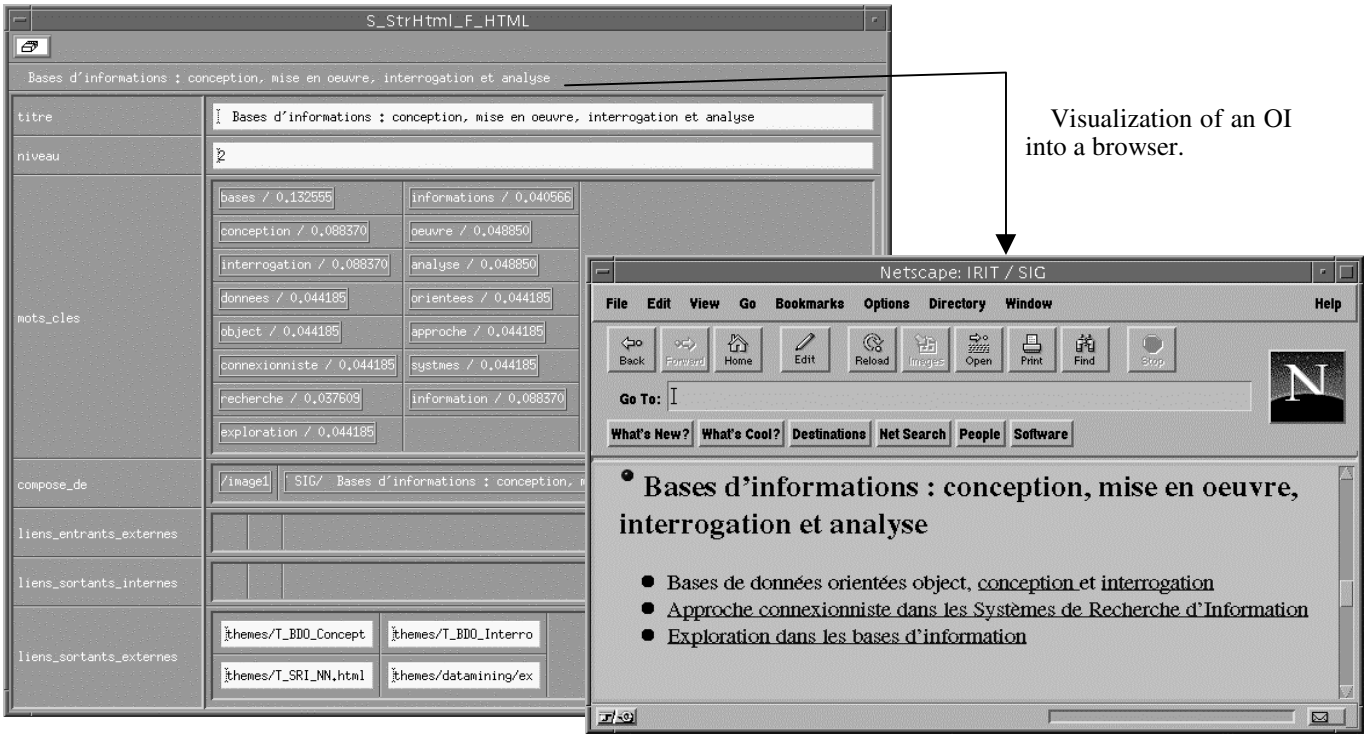



**Figure 7** : IO displaying into the user's browser.

## References


[1] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, J. Siméon, "*Querying Documents in object databases*", International Journal of Digital libraries 1(1):5-19, April 1997.

[2] B. Adelberg, "NoDoSE - A tool for semi-automatically extracting structured and semi-structured data from text documents", pp283-295, proc. of ACM SIGMOD, 1998.

[3] http://www.3.com

[4] G. Arocena, A. Mendelzon, "WebOQL: Restructuring Documents, Databases and Webs", Proc. of 14th. International Conf. on Data Engineering (ICDE 98), 1998.

[5] T. Berners-Lee, W. Connoly, "Hypertext Markup Language-2.0", Internet Network Working Group RFC 1866, MIT/W3C, Nov. 1995.

[6] S. Chakrabarti, B.E. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, "Spectral filtering for resource discovery", pp 13-21, SIGIR Workshop W1 Hypertext Information Retrieval for the web, 1998.

[7] V. Christophides, S. Abiteboul, S. Cluet, M. Scholl, "From structured documents to novel query facilities", pp 313-324, proceedings of ACM SIGMOD, 1994.

[8] M-L. Corral, J. Mothe, , "How to retrieve and display long structured documents ?", pp 10-19, Proceedings of Basque International Workshop BIWIT, 1995.

[9] L. Faulstich, M. Spiliopoulou, V. Linnemann, "WIND : a warehouse for Internet data", pp 169-183, Advances in Databases, 15th British National Conf. on Databases, BNCOD, 1997.

[10] F. Fourel, P. Mulhem, M.F. Bruandet, "A generic framework for structured document access", pp 521-530, DEXA, Vienne Austria,.1998.

[11] G. Gardarin, S. Yoon, "HyWeb : un système d'interrogation orienté objet pour le Web", pp205-224, Bases de Données Avancées, BDA, 1996.

[12] Norme ISO 8879, Information Processing -Text and Office Systems- Standard Generalized Markup Language (SGML), Oct. 1986.

[13] T. Koch, "Metadata and Dublin Core: Introduction", Lund Univ. Library, NetLab, may 1998, http://www.ub2.lu.se/tk/metadata/MDin9612.html

[14] PY Lambolez, JP Queille, JF Voidrot, C. Chrisment, "Exrep : a genric rewriting tool for textual information extraction", Ingénierie des systèmes d'information (ISI), vol. 3 (4) pp 471-486, 1995

[15] R. Light, "Presenting XML", Companion web site SAMS-NET Ed., ISBN: 1575213345, 1997.

[16] Navarro, G. Gonzalo, R. Baeza-Yates, "Proximal Nodes: A model to query document databases by content and structure", pp 400-435, Vol. 15, N°4, ACM Transactions on Information Systems, Oct. 1997.

[17] M. Porter "An algorithmm for Suffix Stripping", Program, Vol. 14(3), pp 130-137, 1980

[18] F. Riahi, "Elaboration automatique d'une Base de données à partir de documents semi-structurés issus du Web", Actes du congrès INFORSID'98, Montpellier, 12-15 Mai 1998, pp. 327-341

[19] G. Salton, E.A. Fox, H. Wu, "Introduction to Modern Information Retrieval", McGraw Hill International Book Company, ISBN 0-07-Y66526-5, 1983.

[20] K. Sparck Jones, "A Statistical Interpretation of Term Specificity and Its Application in Retrieval", Vol 28, N° 1, pp. 11-20, Journal of Documentation, 1972.

[21] M. Theophilactou, M. Lalmas, "A Dempster-Shafer model for document retrieval using noun phrases", pp. 50-

60, Proceedings of 20th Annual Colloquium on IR Research, IRSG, Grenoble, France, 1998.
[22] http://www.rational.com –Rational Software Corporation – Santa Clara USA-
[23] Ross Wilkinson, “Effective Retrieval of Structured Documents”, pp 311-317, International Conf. on Research and Development in Information Retrieval, SIGIR, 1994.
[24] "Extensible Markup Langugage (XML) 1.0 specifications", 3WC recommendation 10/02/98 hpp://www.w3c.org/TR/1998/REC-XML-19980210.
[25] M. Yoshikawa, O. Ichikawa, S. Uemura, “Amalgamating SGML Documents and databases”, Advances in Database Technology, pp259-274, Advances in Database Technology, EDBT, 1996.