Combining the Power of Query Languages and Search Engines for On-line Document and Information Retrieval : The QIRi@D Environment

L. Berti, J.L. Damoiseaux, and E. Murisasco

GECT, Equipe Systèmes d'Information MultiMédia
Université de Toulon et du Var B.P. 132
83957 La Garde cedex, FRANCE
`berti, jld, murisasco@univ-tln.fr`

Abstract. In order to retrieve an item of information on the Web, many search engines have been proposed. They are rarely efficient at the first attempt: the display of results "forces" the user to navigate. In parallel, Web query languages have been developed to avoid these two sequential phases: research then navigation. In this context, the QIRI@D experimental platform, based on the functional programming language SgmlQL, enables both information retrieval and manipulation of distributed semi-structured documents published on a sub-network made up of the sites where QIRI@D is running. It is possible in an unique query to specify criteria to find a document, to filter it, to extract parts of it for building the result. An automatical enrichment of any published document is used to improve the search efficiency.

1 Introduction

The ever-widening growth of the "connected demography" on Internet significantly increases the number, the exchange and the circulation of networked documents. The World Wide Web is a fantastic mine of information with heterogeneous and unequal quality, but lots of documents are not worth of accessing. In a sens, it could be roughly considered as a huge non-administrated and only readable distributed database.

In order to retrieve a specific piece of information, many search engines are proposed ("classical" search engines, multi-search engines, generalized or specialized subject guides...) [17][10][3]. A Web search engine is a database of HTML documents gathered by a robot (program also called wanderer, crawler, spider, or bot...) and tools to generate and search it (with a combination of user-defined keywords and logical operators). Even if their common goal is to localize an item of information by retrieving the documents that match the keywords of the searcher, the way they manage to do it, is significantly different and depends on [15][12][3][2]:

1. the mode of documents collecting (the crop of documents is based on the authors' voluntary service (SubmitIt, Aliweb) and/or automatically generated by the robot which recursively retrieves linked pages and all documents that are referred),
2. the mode of documents indexing (manually or (semi)-automatically, full-text),
3. the pre-processing before the display of results (topics clusters, relevance scores to measure the quality of the match to the search query, duplicate detection),
4. the interface of display (control of the display format, user-defined Preferences, retrieval display options: limiting retrieval by field, manipulating the search set, saving and synthetically visualizing the results).

In spite of their resources, search engines are rarely efficient at the first attempt. End-users may increasingly rely on information professionals for complex searching. Actually, a query must always be specialized or generalized at the second or third attempt. And the original query must be modified in relation to the first list of proposed results by "relevance feedback". The user must adapt to the query syntax of the search engine he's using and to the successive refinements of the queries he has to elaborate. Anyway, the display of results (URL, summary, relevance score...) "forces" the user to navigate.

In parallel, Web query languages [8][9][13] have been developed to avoid these two phases (research then navigation) imposed by robots. Differents elements are essential in such a context:

1. unlike classical query languages which respect a precise data schema, the Web's user does not have any exact knowledge of the data structure within he searches information, neither of the structure of reached documents, query languages have to be adapted to this aspect [1][4][5][6],
2. otherwise, these languages have to consider the hypertextual aspect inherent in the Web structure,
3. finally, it is important to query the structure and/or the content of traversed nodes and to avoid invalid URLs.

To solve these difficulties, some languages propose original and specific solutions. Among them, WebSQL [13] and W3QL [9], SQL-like languages, integrate automatical and supervised navigation into queries, with or without access transparency to index servers for the user. Moreover, W3QL focuses on the extensibility of queries with users programs and UNIX's commands (such as `grep`, `awk`). It is also possible through a query to complete HTML forms encountered during navigation. By means of views, query results can be regularly refreshed. The relevant presentation of the results is another difficulty for query languages. A Datalog-like language, Weblog [8] proposes this original aspect: the restructuration of Web's parts. The user specifies in his query how results have to be presented.

We propose the QIRi@D environment: Quality and Information Retrieval in Electronic Documents, to combine the power of query languages and search engines. QIRi@D is an experimental platform enabling the manipulation of semi-structured documents (SGML and derived) wich are distributed and published on the administrated and controlled QIRi@D sub-network. This sub-network is made up of all the sites where QIRi@D is running. It is possible in a query to specify criteria to find a document, to filter it, to extract parts of it for building the result. The QIRi@D's programming language is SgmlQL [11]. It enables to query the structure and/or the content of documents and is currently extended to embed controlled navigation within a query, which constitutes the essential aspect on the Web.

This paper introduces the QIRi@D environment which proposes another solution for information search through distributed and semi-structured documents. The SgmlQL programming language is shortly recalled in Sect. 2. Sect. 3 details QIRi@D's local and distributed architecture. Sect. 4 and Sect. 5 present the pre-processing of a document and its manipulation within the QIRi@D environment. Conclusions and perspectives are given in Sect. 6.

2 SgmlQL: The Programming Language of QIRi@D

SGML (Standard Generalized Mark up Language)[7] is an international standard for the description of documents. It enables the tagging of document which determines the logical elements of its structure (chapters, sections, paragraphsˇ§). It is also possible to associate some attributes with logical elements to specify them independently of their content. The tagging of document is the only information added to text by means of markers defined in a grammar. This grammar is the core of a "Document Type Definition" (DTD) which represents the generic structure of that document. Notice that tags are not processing indications. So, the tagging makes easier not only the document manipulation but also its circulation and exchange through communication networks.

SgmlQL[11] is an SQL-like language for the manipulation of SGML documents. It is a functional language, an extension of OQL, an object oriented SQL language. SgmlQL has been defined to query SGML documents under any aspect whatever: tree-like structure, attributes, cross-references (hypertextual aspect). Note that SgmlQL directly queries SGML files without using a transcription of documents into a database schema, unlike other languages such as the one proposed by Christophides and al. in [4]. SgmlQL offers operators to browse and transform trees to extract document parts and to build new documents. These operators are embedded in each other such that `select\v{\S}from\v{\S}where`, `replace` or `remove`.

Moreover, SgmlQL[1] manipulates distributed documents through differents sites[14]. In this case, a first URL must be given. In order to briefly present the language, we give two typical queries:

```
Q1. Assignement of the file "Example.html" to the variable mybook.
global mybook = file "http://www.univ-tln.fr/Example.html" ;
```

During the whole session, the variable mybook can be used instead of the URL of file “Example.html”.

```
Q2. What are the Title and the paragraphs of each section of mybook ?
element ANSWER
body :
   select  [first TIT within s, select p from p in every PAR within s]
   from    s in every SEC within mybook;
```

For each section, its first title and all its paragraphs are extracted (the concatenation operator is noted “[ ]”). The result is a new SGML element (tag `answer`), its model content could be such as:

```
<!ELEMENT   Answer          (Title, Par)*               >
<!ELEMENT   Par             (#PCDATA | Bibref)+         >
<!ELEMENT   (Title|Bibref)  (#PCDATA)                   >
```

3 QIRi@D's Architecture

3.1 Overview

QIRi@D is a experimental platform to manage semi-structured, heterogeneous and distributed documents. It mainly provides the searching and the querying of SGML or HTML electronic documents and hyperdocuments. QIRI@D proposes (Fig. 1):

1. the enrichment of each document dedicated to the publishing,
2. the maintenance of the published documents,
3. the consultation of the published documents.

The use of QIRi@D respects the following principle: after the elaboration of a document, the author asks for publishing it in the QIRi@D network. Note that the user must respect a Charter to use the environment QIRi@D (normalized SGML, predefined tags, etc.). After the document publishing, any user who belongs to this network can exploit it. When a document is published, a set of information is extracted, calculated or inferred, to create a new richer document which will be the first target of any search. In a single formalism, user's exploitation combines a www-engine-like search function and the capacity to manipulate the retrieved document through its structure or/and its content. For example,

[1] A complete definition of SgmlQL is available on-line in [16], where the language can be freely downloaded.

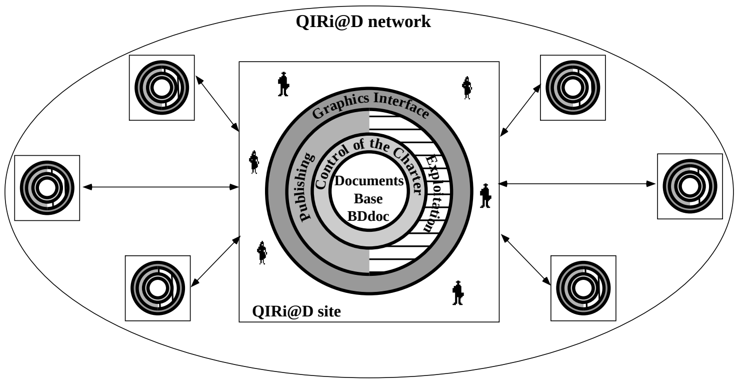

Fig. 1. General Architecture of a QIRi@D site

it's possible to submit a query such as "for each document in St-Malo's server which contain information about buccaneers, extract all images available since january 1997".

QIRi@D runs on a site like a deamon based on the client-server principle. It's made up of tools written in SgmlQL shell and SGML files. The SGML files are built and manipulated by the tools which carry out the following tasks:

1. the publishing of document (enrichment, format control, etc.) ; the users of these services are called *Producers*,
2. the exploitation of document (structure and content manipulation, hypertextual extension, etc.) ; the users of these services are called *Consumers*,
3. the system management.

3.2 BDdoc: The Local Document Base

The local documents base, called BDdoc, is made up of semi-structured documents in various format (SGML, HTML, Latex, RTF) published on this site. As it's shown in Fig. 2, its structure is locally distributed and composed by a set of *Producers* publishing directories (*user-docs*), and a main publishing directory (*system-doc*).

The publishing directory of the producer contains all the documents he wants to publish (*Primary Documents*). This directory is located on Producer login account, but it belongs to QIRi@D which manages it (the Producer can't have access to it). After the Producer asks for publishing (arrow 1 Fig. 2), QIRi@D effectively processes it (arrow 2 Fig. 2). Note, that a publishing operation is local on a QIRi@D site and must be expressed by an adherent of this site. The main publishing directory, managed by QIRi@D, usually contains:

1. all *Enriched Documents* made from the Primary Document processing ; an Enriched Document contains information about its quality and its history and other informations extracted, calculated or inferred from the Primary Document,
2. the index for all *Published Documents* and their localization.

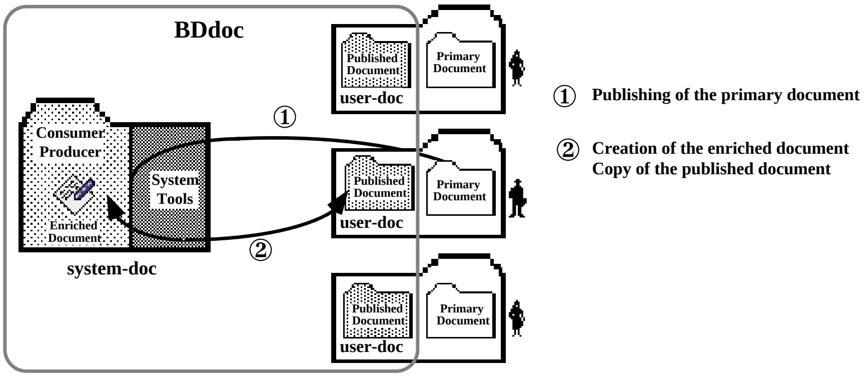

Fig. 2. BDdoc's Architecture

The indexes *Authors_Index* and *Information_Base_Index* (size, quality, modification frequency) won't be detailed in this paper. All these indexes define the structure of the BDdoc and they're stocked in a SGML document (*BDdoc.sgml*), which is an instance of this following DTD (Fig. 3) where the tag *Enriched_Doc* and the tag Form enable to reach through the "Href Attribute", respectively all Enriched Documents, authors information and access statistics (author consultation rate, document consultation rate, ...).

3.3 Distributed Base of the QIRi@D Network

The distributed base is evolutionary and open. It is distributed because it's made up of all Published Documents of all local bases of QIRi@D network (BDdocs) ; a Consumer has access in the same way to every document of the distributed base, even if it is local or distant. Evolutionary because, a Producer, who respects the Charter, can freely modify his Published Documents. It is open because a Producer can create a document which contains links with documents on other sites (adherent or not). However, the QIRi@D base contains a set of documents which are made from the Published Documents. These documents are locally bound to a site, and the Consumer don't have any access to them. They are used by QIRi@D to answer to the Consumer with the reached document.

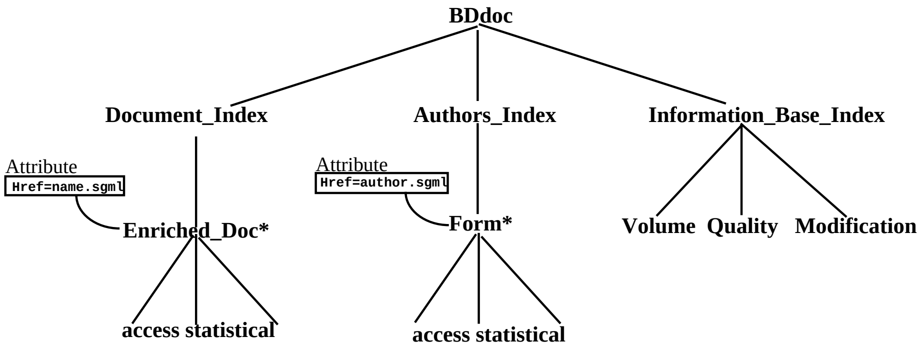

Fig. 3. BDdoc's Structure

4 Document Life Cycle and Its Manipulation in the QIRi@D Environment

The minimal Document Life Cycle in the QIRi@D environment starts with the Primary Document designed and produced by authors. Then, the system builds the corresponding Enriched Document which is the first target of any research query within the whole QIRi@D network. Precisely, the Document Life Cycle (Fig. 4) is based on four steps:

1. the conception of the Primary Document,
2. the local publishing of the Primary Document within its conception site, becoming now Published-Document
3. the creation of the unique associated Enriched Document,
4. the exploitation of the Enriched Document within and by the QIRi@D environment.

4.1 Conception of the Primary Document

A Primary Document is a tagged text in the SGML or HTML format, produced by authors with or without a WYSIWYG editor. In order to be publishable, this document will have to respect the QIRi@D Charter. That is, it will necessarly have to contain specific information integrated in predefined tags (such as authors' names, title and type of the document, ...). The Primary Document could be the root of hyper-documents. The Producer can attach in a separate file the corresponding DTD of the produced Primary Document. A strong constraint for the user to query the structure and the content of a document is to know its DTD.

4.2 Publishing of the Primary Document

When the Primary Document is designed and produced, its Producer can publish it on the site of his QIRi@D membership. If the Primary Document satisfies the

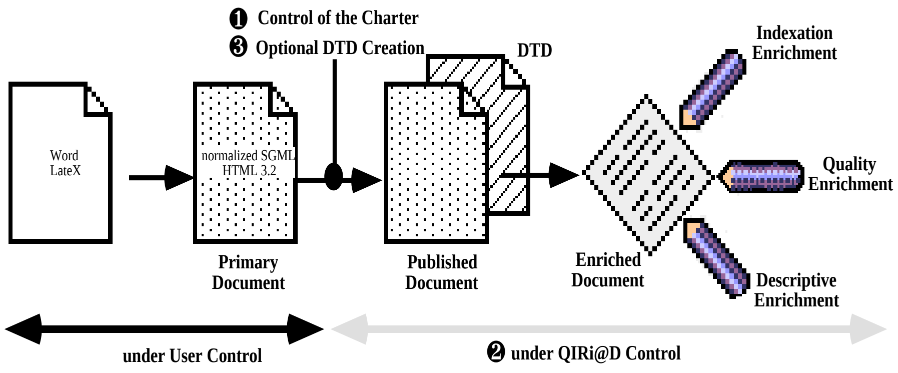

Fig. 4. From Raw Documents to Enriched Documents

set of constraints defined in the Charter, its publishing is possible: it becomes available on the whole QIRi@D network and for every member.

In the case of hyper-documents, the recursive path (following the local links) enables to determine the set of files intended to be published. The external links are not analyzed because the target documents are not local. The publishing of the Primary Document starts with verifying of the (non)-respect of the Charter (cf. 1 Fig. 4), especially about the main points:

1. at least one of the authors must be a QIRi@D member,
2. the tagging format must be respected (version 3.2 for HTML, closing tags are obligatory for SGML...),
3. the document must contain the predefined QIRi@D tags (title, author, keyword, category).

Then:

1. the component file(s) are moved in the user directory of publishing (cf. 2 Fig. 4),
2. the DTD of the Primary Document is automatically constructed (cf. 3 Fig. 4) if it has not been alreadyattached by the Producer,
3. the associated Enriched Document is elaborated,
4. *Information_Base_Index* is updated.

4.3 Creation of the Enriched Document

The Enriched Document is elaborated in adding information about the Published Document and it's an instance of the DTD shown in the Fig. 5which contain both generated information (e.g *DocId*) or extracted information about the Published Document (e.g as *Body*):

1. DocID is a SGML element, its content is an unique identifier affected to the Primary Document when it's published. The associated attributes are query results (cf. query Fig. 5) and store information about physical characteristics of the Primary Document (format, size, date),
2. the Primary-DTD is a SGML element, its attribute Href targets the document containing the DTD of the corresponding Primary Document ; this DTD is essential for the Consumer who will be able to make refined search on the document,
3. Body is a SGML element, its content is information extracted from the content of the Primary Document and which are likely to answer to general queries. Among the elements of its content:
   a) the tag *Category* specifies the category of the document (article, course, ...),
   b) the tag *Keyword* is a list of the keywords of the Primary Document,
   c) the tag *Links* is a list of optional elements which contains for a Published Document the different URLs within the document(*Internal Link*), within the site (*Local Link*) and outside the local site (*External Link*).

Note, any Enriched document refers to its corresponding Published document, but is not its extented copy.

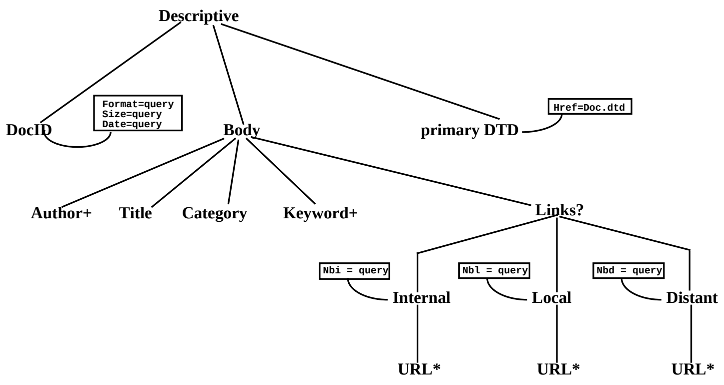

Fig. 5. Structure of an Enriched Document

4.4 Exploitation of an Enriched Document by QIRi@D

The QIRi@D environment proposes the twofold exploitation of Enriched Documents:

1. the consultation: a consumer queries the QIRi@D network from its membership site. The query is locally processed and if it's not sufficient (if the query doesn't concern the local site, or if the site/domain is unknown), it's

sub-processed through of the QIRi@D network. Each site evaluates the query on all local base of Enriched Documents. If the answer to the query is found in the Enriched document, the corresponding Published Document is not reached. If the answer to the query is not found in the Enriched document, the corresponding Published Document is reached, thanks to the DocID tag of the Descriptive tag (cf. Fig. 5).

2. the maintenance of Published Documents by the system by mean of specific operators (update, delete). When a Producer wants to update a document that he has already published, he must modify the Primary Document and must publish it again with the same name: it is a new publishing. The associated Enriched Document is updated excluding its identifier. Its trace is updated. Only Producer can delete his own Published Documents. Then, QIRi@D deletes the associated Enriched Documents, updates informations of the local base (BDdoc), and finally deletes the Published Documents and their possible linked files (if they are hypertexts).

5 Consumer's Document Manipulation

This section illustrates a Consumer manipulation with the SgmlQL language. Assume that a Consumer searches all documents which deal with HTML on the Toulon's QIRi@D site. This Consumer wants to extract the title and the authors of each found document and expresses it with query Q1:

```
Q1: Title and Authors of documents dealing with HTML on Toulon's site
select  [first TIT within d , every AUT within d]                    (1)
from    d in (every DOC within domain "univ-tln")                    (2)
where   exists c in (every KEYWORD within d) : c match "HTML"        (3)
```

Line (2) builds the list of all the documents found on Toulon's site, and defines the variable d to browse that list. Line (3) checks, for each value of d – that is each document found – , whether its keywords match "HTML". Line (1) extracts for each value of d, the title of the current document and its authors. Actually, for an effective execution of that query, Toulon's QIRi@D environment locally applies the query Q1bis derived from Q1 on each Enriched Document of that site.

```
Q1bis : QIRi@D translation of Q1 query
select  [first TIT within file d->HREF, every AUT within file d->HREF] (1)
from    d in (every ENRICHED_DOC  within file "bddoc.sgml")            (2)
where   exists c in (every KEYWORD within file d->HREF):c match "HTML" (3)
```

The bddoc.sgml (cf. Fig. 3) document contains a set of tags ENRICHED_DOC (line 2). For each one of these tags, the attribute HREF is defined and its value is an URL of an Enriched Document. The content of each Enriched Document is reached and its keywords are compared with the string "HTML" (line 3). The Consumer's query (line 1) extracts information through each one of these documents.

The four steps of Q1's evaluation are:

1. Q1 is locally typed by the consumer on his membership QIRi@D site,
2. the local SgmlQL interpreter analyses, optimises this query, and sends it to the concerned site(s) for its execution, in our example the "univ-tln" domain,
3. Q1 is then transformed into Q1bis on Toulon's site by the system,
4. the result of Q1bis is sent to the consumer's site.

6 Conclusion and Perspectives

We described in this paper the QIRi@D experimental platform for the management and the exploitation of semi-structured and distributed documents on QIRi@D network. Its implementation is in progress at present time. The system is entirely built upon the SgmlQL query language. The user is able, with a unique query, to use the same web-engines-like search function and to manipulate the retrieved and reached documents thanks to the automatical enrichment of documents elaborated by the QIRi@D system (about the structure and/or the content of documents, their hypertextual extension, their access history...).

In order to introduce new dimensions in the search and in the exploitation of on-line documents, our ongoing works focus the notion of enrichment concerning:

1. differents types of hypertextual links,
2. the data and document quality ; quality tags are integrated:
   a) for extending the notion of relevancy-ranking[17][3] proposed by the search engines,
   b) for personal and adaptative filtering of data and documents (user acquisition profile),

Moreover, the Producer will be able to add meta-information into his Primary Document during the conception step, to improve the indexation and the intrinsic quality of his document.

The main application of our environment is dedicated to Intranet applications.

References

1. S. Abiteboul, D. Quass, J. McHugh, J. Widow, J. Wiener,
   The lorel query language for semi-structured data,
   http://www.db.stanford.edu
2. L. Barlow, The Spider's Apprentice,
   http://www.monash.com/
3. K.Campbell,
   Tips on Popular Serach Engines,
   http://www.hamline.edu/library/bush/handouts/slahandout.html

4. V. Christophides, S. Abiteboul, S. Cluet, M. Schol,
From structured Documents to Novel Query Facilities,
Proceedings of ACM-SIGMOD'94, Minneapolis (US), May 1994
5. C. Clifton, H. Garcia-Molina, D. Bloom,
Hyperfile : a data and query model for documents,
VLDB journal , n4, pp. 45-86, 1995
6. G. Gardarin, S. Yoon,
HyWeb : un système d'interrogation pour le Web,
Bases de Données Avancées (BDA' 96), Cassis 1996
7. C. F. Goldfarb,
The SGML Handbook,
Oxford Clarendon Press, 1990
8. L. V. S. Lakshmanan, F. Sadri, I. N. Subramanian,
A declarative Language for querying and restructuring the Web,
Proceedings of the Int. Workshop on Research Issues in Data Engineering (RIDE'96), New Orleans, 1996
9. D. Konopnicki, O. Shmueli,
W3QS, a query system for the world-wide Web,
Proceedings of the 21st VLDB Conference (VLDB'95), Zurich, Suisse 95
10. M. Koster,
The Web Robots Pages,
http://info.webcrawler.com/mak/projects/robots/robots.html
11. J. Le Maitre, E. Murisasco, M. Rolbert,
SgmlQL, a language for querying SGML documents Proceedings of the 4th European Conference on Information Systems (ECIS'96), Lisbon-Portugal, 1996
12. J. Liu,
Understanding WWW Search Tools,
http://www.indiana.edu/ librcsd/search/
13. A. O. Mendelzon, G.A. Mihaila, T. Milo,
Querying the World Wide Web,
Journal Digital Libraries, vol. 1, no. 1, 1997
14. P. Ramadour, E. Murisasco, J. Le Maitre ,
Optimisation de requêtes mono-site et multi-sites,
Rapport Interne no. 97-1, GECT, octobre 1997
15. W. D. Sullivan,
A Webmaster's Guide To Search Engines,
http://searchenginewatch.com/wgtse.htm
16. SgmlQL Manual,
http://www.lpl.univ-aix.fr/projects/multext/mtsgmlql.html
17. R. Tyner,
Sink or Swim : Internet Search Tools & Techniques,
http://oksw01.okanagan.bc.ca/libr/connect96/search.htm