INTEGRATING THE ACCESS TO DOCUMENTAL DATABASES ON THE WEB

Nieves R. Brisaboa María J. Durán Charo Lalín Eva L. Iglesias
Juan R. López José R. Paramá Ángeles S. Places Miguel R. Penabad

Laboratorio de Bases de Datos. Facultade de Informática
Universidade da Coruña, Spain

brisaboa@udc.es

ABSTRACT

This work presents a proposal of architecture for a system designed to provide a uniform access through Internet to different distributed and heterogeneous documental databases, which in our case will store antique Spanish documents and information about them. Our main goal is to offer an intuitive interface, designed using the ideas of Virtual Book and Virtual Library metaphors. This interface should let the user ask complex queries with a total transparency about where the databases are, which ones are needed, and how their conceptual scheme, DBMS and query languages are.

The architecture we propose is composed of four well-isolated levels, thus easily allowing for the scalability of the system: the first one is a unique user interface to access all databases; a second level that distributes the queries and integrates the answers; a third one that deals directly with the databases; and the databases themselves. So far we have developed an interface to access a database that stores a corpus of Spanish Emblem Literature.

We propose the use of XML (eXtended Markup Language) to define a Formal Query Language (XML-FQL) and a Formal Answer Language (XML-FAL) that will provide the needed independence between the databases and the other layers of the system. For the query interface, Virtual Book and Virtual Library metaphors are used to allow the users to express sophisticated queries. In addition, *Bounded Natural Language* sentences are used in order to provide a flexible, powerful and easy to use interface.

1. INTRODUCTION AND MOTIVATION

Nowadays it is possible to access a growing number of document archives through Internet. Some of them are catalogues from libraries, stored using different types of DBMSs, the most common of them the Relational Database Management Systems (RDBMSs), used to store just bibliographical information. There are also collections of documents that are stored either in plain (ASCII) text or SGML (usually HTML) documents. In fact, Internet itself could be seen as an enormous documental database whose documents are stored in HTML format. It is also common to find sets of documents stored as digitized images of the pages from the books, often labelled with key words that are useful to perform searches among the documents.

The two main problems we have to deal with if we want to take advantage of such a source of information are:

- The databases are documental, storing both data and text.
- The databases are distributed, residing in different servers, and may have different DBMSs and conceptual schemes.

Concerning the first problem, there have recently appeared systems that take advantage of the features of the new commercial database management systems, which store both bibliographical information and the text of the documents, partially of fully transcribed. These DBMSs allow storing text as an attribute. For example, Oracle incorporates a new module, *ConText* [12], which offers the possibility of querying a database by "normal" data (the traditional relational data) as well as by the words that appear in the transcribed texts. This is made by extending the SQL language, adding a “Contains” clause in which the words that must appear in the documents are specified, possibly including logical connectors to establish relationships among them. Using this clause, a single SQL statement integrates constraints about both normal attributes and the words that appear in the new "Text" attributes. *ConText* also offers lemmatisation and synonyms search capabilities, but so far only available for documents written in English. Similar capabilities can be found in *Excalibur*, the text retrieval module used by Informix [15].

Both *ConText* and *Excalibur* offer interesting solutions to query relational as well as textual information. However, they do not implement (so far) all the potential offered by the different text retrieval techniques [1, 2].

A different issue concerning the problem of querying documental information is the design of a good user interface. In a documental database it can be found

bibliographical information such as author, title, publishing date, etc., but also documental information like transcribed texts or key words that define the main topics of the document. Nevertheless, the latter is seldom used to query the databases. "Topic queries" that consider this documental information would be very useful, but they are not commonly implemented in query systems, although there are some of them, like the Guttemberg project [13] that offers a topic search. The rest of them offers "boolean search", that is, the user enters several words connected by boolean operators, such as "and" and "or". However, the use of boolean queries has three main problems:

- "And" and "Or" are confusing, counterintuitive and not easy to use, especially for naive users.
- The "Or" operator restricts the query too few (just one of the typed words is enough for a document to be retrieved) and the "And" operator restricts too much (all of the words must be present).
- A document is retrieved or it is not, but it is not possible to sort (rank) the documents according to their relevance.

In the architecture we propose in this paper, the logical (boolean) connectors are avoided except for formal data (bibliographical data), and we introduce the use of Text Retrieval (TR) techniques [17, 18, 19, 20] to base the search in the documents main topics. Such techniques will yield a better accuracy in the retrieved documents, and it will allow the interface to sort them according to their relevance to the query.

The second mentioned problem was dealing with heterogeneous and distributed databases. The corpus of documents accessible via Internet are usually managed by different DBMSs and have different conceptual schemes, and they often reside in different servers at different locations. On the other hand, the representation of documents varies among servers, and so does the query languages that can be used to perform searches. This is a strong handicap when trying to integrate several systems in a unique point (a unique user interface to access them all).

The main implication of these aspects are the following:

- The user interfaces used to perform searches are different for each corpus, therefore the users usually need some kind of training to be able to use the interface, to understand the database schema, etc.
- The databases can be disperse among different servers and geographical locations, making it necessary to know their address in order to access them.

All these restrictions limit the quantity and quality of the information that could be extracted from the documents, because they difficult the access. Currently it is often necessary to access the different corpus separately and issue the same query (using different interfaces) to all of them to find the desired information.

In order to try to solve these problems, it appears the necessity to develop systems that permit a uniform access to all databases needed to solve a query [16]. The main idea is to design a system that unifies through a unique point all databases, having an interface that hides the differences between the databases schemes as well as their geographical dispersion. Such a system ought to match two fundamental requirements in its design:

- *Its Architecture must be scaleable*, that is, adding new databases with new documents (maybe remotely located, supported by different DBMSs and having different conceptual schemes) must be an easy, quasi-automatic procedure.
- *The User Interface must be friendly*, easy to use, powerful and flexible to match users requirements. It is a crucial aspect, because if the users do not find it accurate, they will not use the system.

The first requirement leads to the necessity of developing two formal languages: a Formal Query Language that is used to describe the queries, and a Formal Answer Server that is used to represent the documents. By using them, the process of integration of several systems is almost immediate, because all of them share the same languages to communicate. Only at the low level, when dealing directly with the documents and databases, a translation from the internal representation and the formal languages (and vice versa) is needed.

The user interface we propose will not be only useful for issuing queries. It will serve as a (unique) start point where all investigators can share their knowledge through different services, such as discussion forums, bulletin boards, etc. The queries and answers will be shown to the user by means of a Book and Library metaphors that makes it intuitive and easy to use. For sophisticated queries that search documents by their topics we will use *Bounded Natural Language* sentences.

The rest of this paper is distributed as follows. Section 2 describes the full system architecture. Section 3 describes the user interface along with the concepts of Book and Library metaphors. Two formal (query and answer) languages are described in Section 4. The last section contains our conclusions.

2. SYSTEM ARCHITECTURE

Designing and implementing a like the one described in the previous section requires a deep analysis and a well-designed architecture. Following the first of the two guidelines described above, the architecture we propose in this paper was designed to be scaleable. Therefore, it will fulfil the aim of adding new databases to our system in an easy way and without affecting the working system.

The proposed architecture is composed of 4 layers, as shown in Fig. 1.

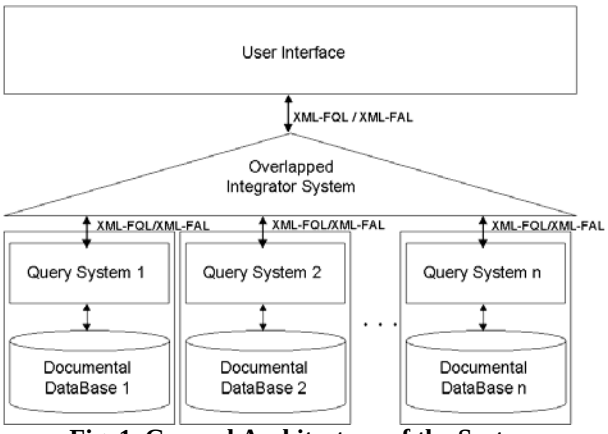

Fig. 1. General Architecture of the System.

We shall describe its layers in a "bottom-up" basis, from the databases to the user interface, but the general behaviour of the system is the following: The user specifies a query using the User Interface; the query is sent to the Overlapped Integrator System, which distributes the query among the Query Systems accessing the databases. Each database answers the query with the set of document that satisfy the constraints imposed in the query. The answers coming from each database are returned to the Overlapped Integrator System, who integrates them and shows them to the user by means of the User Interface. Fig. 2 shows a more detailed view of this architecture.

In addition, two new formal languages are being defined to establish a communication between these layers. They are XML-FQL (Formal Query Language), which is used to build the queries, and XML-FAL (Formal Answer Language), to represent the answers (documents).

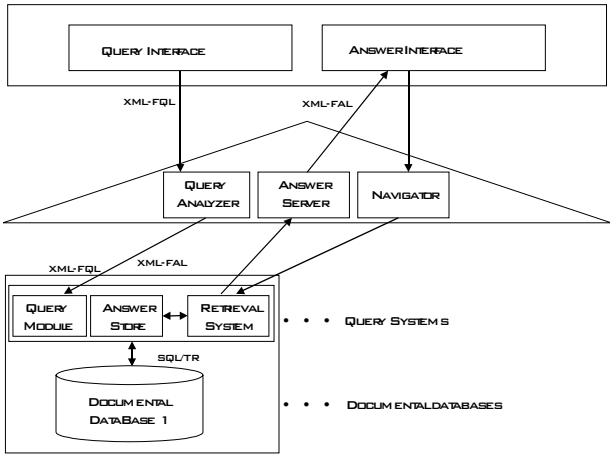

Fig. 2. A more detailed view of the architecture

2.1 Layer 4: Heterogeneous, distributed documental databases

The databases that will be accessed using the proposed system have some common characteristics. They are:

- **Heterogeneous**: The databases can be managed by different DBMSs with different data models (relational, object-relational, etc.), and they can have different schemes.
- **Distributed:** the databases will be distributed, located in different information servers and locations.
- **Documental**: They store (formal, relational) data about documents, but can also store the text of the documents in plain ASCII or using SGML. They can also store the digitized images of the original editions of the documents, which is especially interesting when the documents are antique. Usually, index structures to permit Text Retrieval queries are also included in the databases.

2.2 Layer 3: Query system

Every Query System will be associated to a particular database, and its job is to search for the documents solicited in the query. Although all the Query Systems perform similar tasks, they need to be adapted to the specific associated database, depending on its scheme and DBMS. The following modules, as shown in Fig. 2, conform each Query System:

- **Query Module:** It takes the part of the query from the *Overlapped Integrator System* that concerns its associate database, and translates the XML-FQL query dividing it into two parts: SQL for the data that describes the documents and Text Retrieval for the documents themselves. Using an intelligent combination of SQL and TR, the Query System can retrieve the documents that match the restriction imposed by the user in the *Query Interface*, whether these restrictions imply formal (usually bibliographic) data about the documents, or the subject or topic the documents are about.
- **Answer Store:** It stores the identifiers of the documents retrieved by the Query Module. These identifiers can be sorted by any specific criteria stated in the query or by the relevance of the document with respect to the query. For each document it is also necessary to store an identifier of the session or user who made the query, to avoid conflicts when querying concurrently (it applies a philosophy similar to the Z39.50 protocol [14]). The Answer Store tells the *Answer Server* (in the *Overlapped Integrator System*) how many documents were retrieved by the execution of the query.

- **Retrieval System:** This module retrieves all the data associated to each document (i.e., images, texts and data) using the document and user/session identifiers given by the Answer Store. This module accepts requests from the Navigator (identifying the desired document) and transmits the answers (text, images and data of the documents) to the Answer Server, both modules belonging to the Overlapped Integrator System. The main advantage of the use of this module is that there is no need to send all the information (all the obtained documents) at once, but only on demand of a chosen document.

2.3 Layer 2: Overlapped Integrator System

The main task of the Overlapped Integrator System is to act as a bridge between the User Interface and the Query Systems of the databases that are needed to fulfil a query. It analyses the query (that comes in XML-FQL format) and produces a set of subqueries, which are distributed among the Query Systems of the databases involved in it. When the answer comes from them (in XML-FAL), the Overlapped Integrator System integrates the answers from all of them, so the information can be presented to the user in a convenient way. Besides, it maintains the session so the user can navigate through the obtained documents. It is formed by the following modules:

- The **Query Analyser**, which performs 2 tasks: it decides which of the databases are involved in the query (maybe not all of them are), and it redirects the part of the global query that affects a given database to its *Query Module*.

  Note that the unique used language is XML-FQL. The reason for that is that at this level the conceptual model of the database and the DBMS that manages it is unknown. Therefore, it is a good way to preserve the "physical" independence and thus improving the scalability of the system.

- The **Answer Server** integrates the answers that come from the different databases and generates the main page for the *Answer Interface*. It takes the number of obtained documents from the *Answer Store*, and sends to the *Answer Interface* the information necessary to build the page of the virtual book where the user can navigate through the set of documents that has been retrieved.

  The Answer Server uses the Formal Answer Language XML-FAL to communicate with both the Answer Interface (in Layer 1) and the Answer Store (in Layer 3), in order to gain independence from them.

- The **Navigator** maintains the current user session and acts as a bridge communicating the answer interface with the query system. Following the commands issued by the user in the answer page, it navigates through the documents, sending commands to the appropriate *Query System*, which obtains the documents using the document and session identifiers provided by the *Answer Store*. By using this technique, we avoid the transmission of all the information about all documents at the same time, thus improving the response time. This is a fundamental aspect of the system, because the transmission speed of Internet is still low, and sending all the information at once would make the system almost unusable.

2.4 Layer 1: User Interface

The user interface, entirely developed using Java, offers a common start point to access several services. It permits a unified access through Internet to all databases currently integrated into the system, making the search process and the number and location of the databases transparent to the users. It also offers a *join section* for any user to become a member of our Virtual Library and then gain access to other services. As this interface is one of the main contributions of this paper, the next section is devoted to describe it.

3. THE USER INTERFACE

The user interface, from the viewpoint of the architecture described below, is composed of two modules, the *Query Interface* and the *Answer Interface*. Besides, there are other services available to "registered users" or "members". The main page uses a "library metaphor". It resembles a real library, both in its aspect and its functions, containing the different sections. Generally, these sections can be divided into two types:

- Services available for members.
- Query/Answer Interface.

Fig. 3 shows the main page, resembling a library. It provides direct access to the different sections available. As it can be seen, the search interfaces can be accessed to perform either a global search among all the available corpus or a specific search, first choosing the corpus. This picture shows our prototype, which already has some services (join, discussion forum and bulletin board) and the query/answer interface for one of the corpus we are working with (Spanish Emblem Literature).

3.1 Services available to members

There are different services that are offered to the researchers so they can share their knowledge, inform about incoming events, start discussions about some specific subject, etc. However, before any user can have access to these services, he or she must become a member of the virtual library. This is made through the *Join Section*. The user will be provided with a login name and a password (as a sort of member ID for a real library).

This will grant the user access to other services of the library, and the ability to store his/hers queries for later use.

The most important proposed sections for the Virtual Library, to serve as a forum where the members can interchange ideas, are:

- *Discussion Forum*: In a similar fashion as a common "chat", but only available to members, this forum allows for a real-time, written communication.

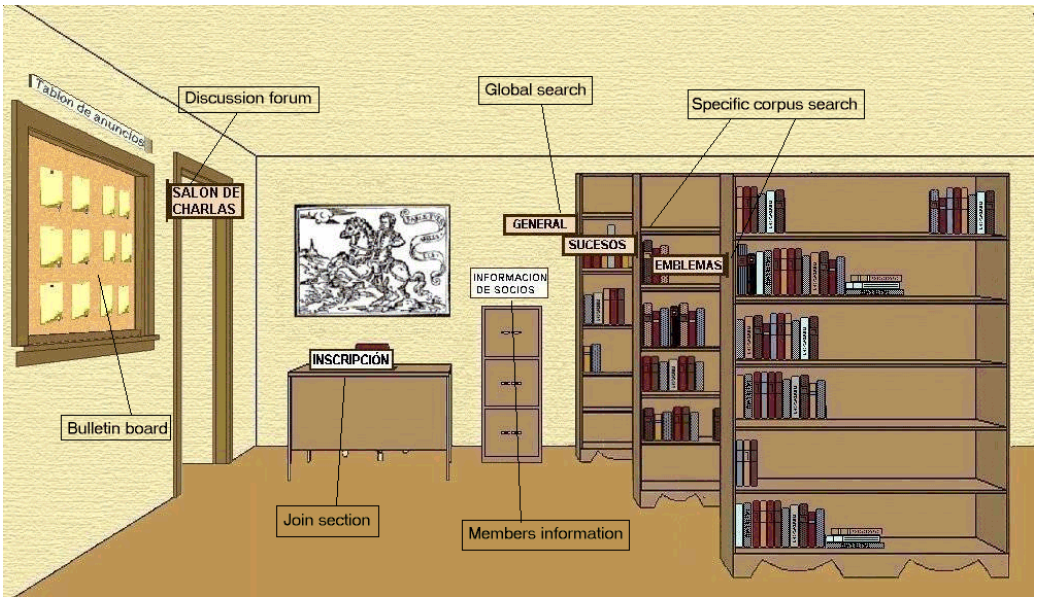

Fig. 3. Main page of the interface using a "library metaphor"

- *Bulletin Board*: Its functions are similar to those of the News groups. Any member can leave a question to be solved by any other member, but not in real time. Besides, it is the best place to leave news and advertisements of symposiums or other events that may be of interest for the community.

- *Information about members*: It shows a list of members, with information about them, such as name, address, affiliation, phone, personal Web page or e-mail.

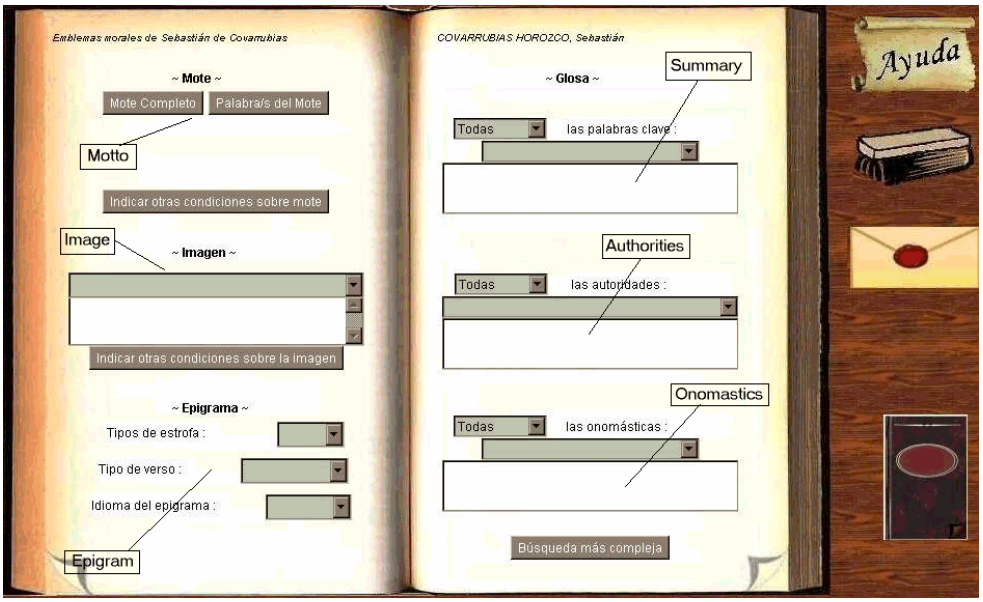

Fig. 4 The virtual book resembles an emblem

3.2 Query Interface

This section will be used to perform searches for documents, either based on bibliographical data such as author, date, etc., or based on the documents' topics. Furthermore, the user will be able to select the corpus of documents that will be involved in the search.

After selecting either a global search or a specific corpus, the cover of a book is presented to the user, where she can choose an author and/or title. The "all books in the corpus" option is also available if the user is not interested in any specific book or author. In this case, the searches will cover all the books in the corpus.

The next step uses a "book metaphor". It shows an image of an open book (a *Virtual Book*) with the format of the books of the corpus, as shown in Fig. 4, where the chosen corpus was Spanish Emblem Books. As it can be seen, it resembles an opened Emblem book (the objects on the right side are Help, Clear, Send, and Back "buttons"). Note that the position and meaning of each part of the book are considered. In the example shown in the figure, the motto, image, epigram and the commentary are in the same place, so any user would know what each part is, just looking at it. If no corpus is selected, then a general format of a book will be used.

Using this book metaphor, users have just to fill in the appropriate information to obtain what they want. There are also sections in this virtual book that open new pages to specify more complex queries, such as those based in the documents topics. This queries will use text retrieval techniques to retrieve the appropriate documents. An important aspect of this part of the query interface is the ability to express the queries using a *Bounded Natural Language*. The main idea, as shown in Fig. 5 (we have only so far the Spanish version), is to provide the user with a set of natural language sentences with gaps that the user will fill using restricted sets of words (available to the user in a list box). In this way, the user can express in natural language her query, just by selecting sentences and adapting them to express what she desires.

There are two main types of gaps in a bounded natural language sentence:

- **Condition gaps,** to be filled with words that restrict the data that will be obtained. The user can type the words in the edit field but, or the words can also be chosen from a list box that contains the valid ones.
- **Modifiers:** These gaps are used to give different relevance to a sentence, or to relate the condition expressed by a sentence with the rest of the used sentences, avoiding the use of logical connectors. This is achieved by assigning a *weight* that depends on the modifier, which is usually chosen from a list box.

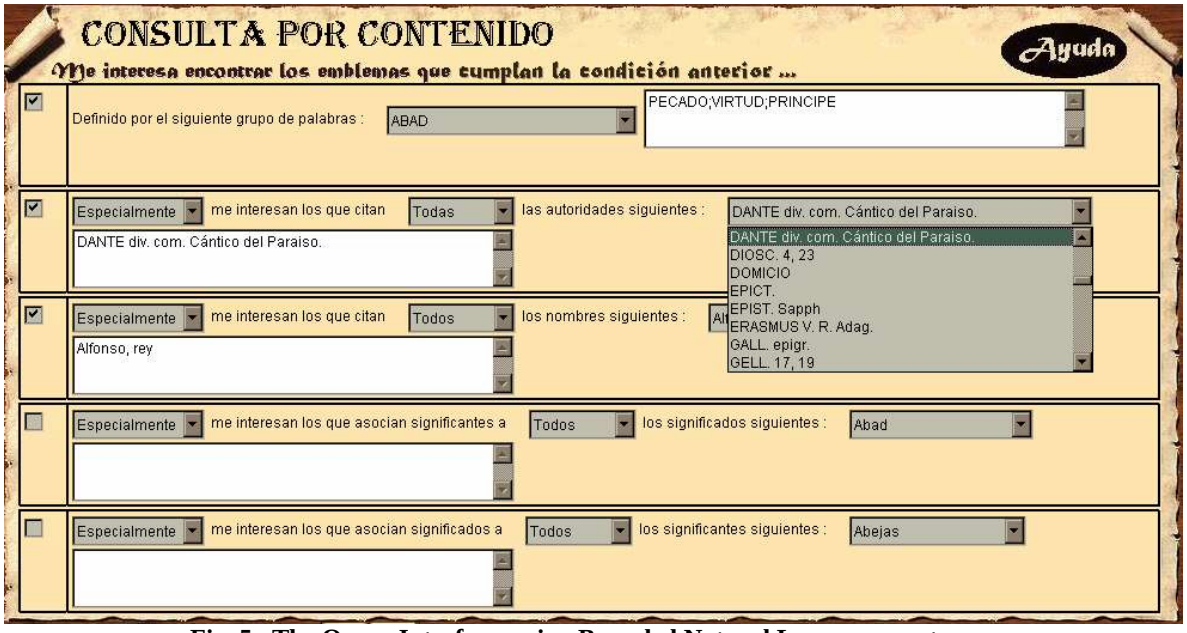

Fig. 5. The Query Interface, using Bounded Natural Language sentences

Following with the general process of issuing a query, once the desired sentences are marked and filled with the words that express it, the query is built. The process of building a query translates the restrictions imposed by the user into a sentence in a formal language, XML-FQL (Formal Query Language). The query will be passed down to the Overlapped Integrator System, and the answers coming from this layer (in XML-FAL, the Formal Answer Language) will be translated into the appropriate language to be displayed in a Web page.

3.3 Answer Interface

The *Answer Interface* will show the documents matched by a query, using again a book metaphor. In an intuitive way the user can go through the pages of the Virtual Book, being able to see the pages that correspond to documents on a given subject or written by an author. For every document, the original digitized page can be displayed if it is available in the database. We show a proposal of this Virtual Book in Fig. 6, which has been already developed for one corpus, the Spanish Emblem Literature.

Internally, the answer interface obtains the information from the Overlapped Integrator System in XML-FAL format, and translates it into the appropriate language to be viewed by the users.

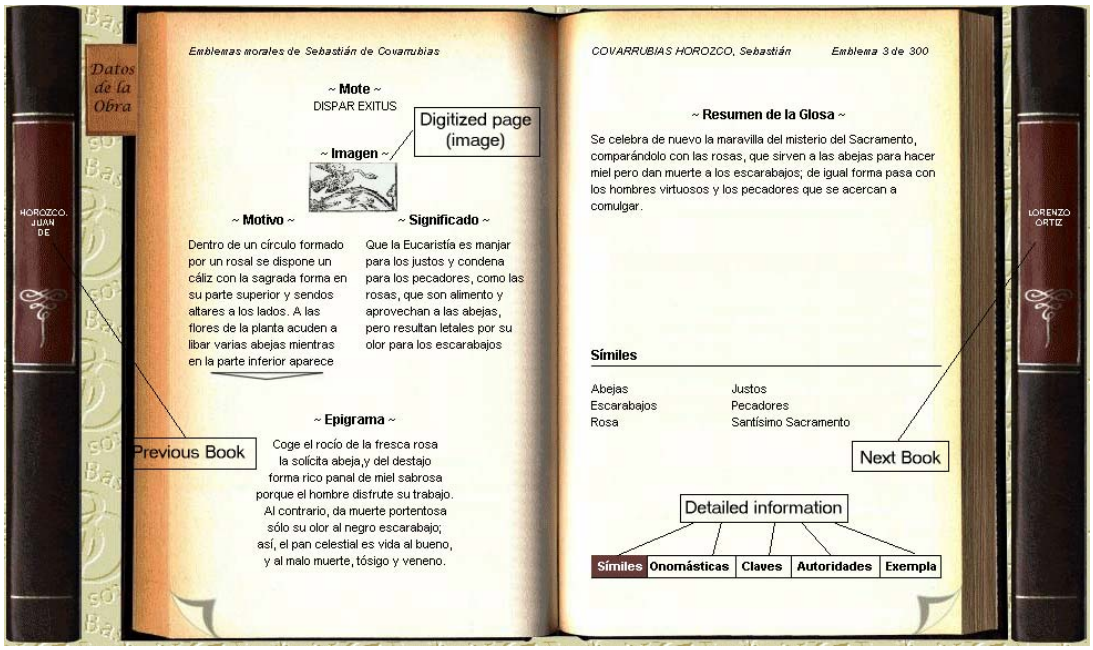

Fig. 6. Virtual Book

4. FORMAL QUERY AND ANSWER LANGUAGES

The motivation that lead us to the idea of having to define two intermediate formal languages is that the system will deal with different DBMSs and different representations of documents. If the system were to use directly the document description languages or the query languages for every different system involved, it would be an almost impossible task to integrate different systems in order to offer a unique access point. There is a need for a formal language to describe the queries and the documents, so some level of independence can be gained.

As shown in the previous sections, we propose the definition of two formal languages:

- **XML-FQL** (Formal Query Language)
- **XML-FAL** (Formal Answer Language)

Both of them are built using XML, which is an appropriate metalanguage, especially defined to its use in Web environments. It has the advantage of appearing to be an incoming *de facto* standard. The main advantage of the use of these two formal languages is the independence they provide.

As for the architecture we propose in this paper, using XML-FQL/XML-FAL we can achieve independence between the Overlapped Integrator System and:

- The Query Interface: It is possible to change the Query Interface without affecting anything, provided that the new interface uses XML-FQL to formulate the query and expects the answers coming in XML-FAL.
- The Query Modules, thus giving independence from the databases, their conceptual models and DBMSs.

5. CONCLUSIONS

The design of an architecture as the on shown in this paper represents a challenge in three different areas: The design of a friendly user interface; the integration of distributed, heterogeneous documental databases; and the use of Text Retrieval techniques for documents written in Spanish. Besides, two formal languages (XML-FQL and

XML-FAL) have to be defined in order to provide several levels of independence.

Undoubtedly, this system will have a great deal of usefulness for the community of Internet users. All non-informatic users will see it as a virtual replica of a library, a familiar environment, but having the advantage of a broader range of possible queries. They will be able to continue using common bibliographic queries, but they will be also able to ask for documents basing their queries on their subjects. Besides, there is no need to know the underlying schema of the databases (or even their number or location), because queries are built using (bounded) natural language.

We are so far dealing with 2 documental databases [9, 10]. The query interface using bounded natural language is working with one of them [9], and we are working in the design of the global architecture described in this paper, in order to incorporate new databases, integrating them with the two available now.

REFERENCES

1. R. Baeza-Yates, W. Cunto, U. Manber and S. Wu. Proximity matching using fixed-queries trees. Proceedings of CPM'94, LNCS 807, pp. 198-212. (1994)
2. R. Baeza-Yates, G. Navarro. A faster algorithm for approximate string matching. Proceedings of P CPM'96, LNCS 1075, pp 1-8 (1997)
3. S.Biagioni, J. Borbinha, R. Ferber, P. Hansen, S. Kapidakis, L. Kovaks, F. Roos, A.M. Vercoustre. The ERCIM Technical Reference Digital Library. Second European Conference on Research and Advanced Technology for Digital Libraries (Sept. 1998)
4. C. Bowman, P. Danzing, D. Hardy, U. Manber, M. Schwartz. Harvest: A Scalable, customizable discovery and access system. 1994. CU-CS-732-94.
5. Brisaboa, N.R., Hernández, H.J., Iglesias, E.L., López, J.R., Paramá,J.R. Penabad,M.R. " An emblem Literature Database On Internet". Proceedings of IDPT'98. Berlin. 1998.
6. J. Davis. Creating a Networked Computer Science Technical Report Library. D-Lib Magazine (Sept. 1995)
7. J. Davis, C. Lagoze. The Networked Computer Science Technical Report Library. D-Lib Magazine, Julio 1996. Cornell University.
8. P. de la Fuente, J. Vegas. Una Biblioteca Digital: Los manuscritos de la Biblioteca Histórica de Santa Cruz. BUCLE2. León. Noviembre (1998)
9. http://rosalia.dc.fi.udc.es/cicyt/
10. http://rosalia.dc.fi.udc.es/XUGA10504A96/
11. http://sunsite.Berkeley.EDU/PhiloBiblon/phhm.html
12. http://www.oracle.com.
13. http://www.promo.net
14. Information Retrieval (Z39.50): Application Service Definition and Protocol Specification. 1995. http://lcweb.loc.gov/z3950/agency.
15. Lorenzo, E.L., Gallego, P., Brisaboa, N. R., Amenedo, D.L., Penabad, M. R., and López, J. R. (1999). "Recuperación de textos en Sistemas de Gestión de BD Relacionales". In proceedings of Segundo Encuentro de Computación 1999 (ENC'99). 12-15 September. Pachuca, Hidalgo (México) (Accepted for publication).
16. OBSERVER: An approach for Query Processing in Global Information Systems bases on Interoperation across Pre-existing Ontologies. Ph.D. Thesis. Univ. Zaragoza (1998)
17. Salton, G Automatic information Organization and Retrieval. McGraw-hill (1968)
18. Salton, G. Introduction to Modern Information Retrieval McGraw-Hill (1983)
19. Salton, G. Automatic Text Processing. The Transformation, Analysis and Retrieval of Information by Computer. Addison-Wesley (1989)
20. Salton, G. The SMART Document Retrieval Project, Proceedings of SIGIR'91, ACM Press (1991)