Non-Materialized Object Views of Web datasources

Zoé Lacroix
Data Logic, a division of Gene Logic Inc.
2001 Center St, Suite 600
Berkeley CA 94704, USA
zoe@genelogic.com

ABSTRACT

Web datasources are mostly textual and often provide full-text search options to access data. However accessing Web sources filling forms, browsing and reading is tedious. Most users are used to access and manipulate data with a rich query language and expect to query Web datasources the same way. To satisfy users, we propose an approach based on a non-materialized object view of Web datasources. Each query against the object view generates calls onto the Web datasource. Retrieved documents are parsed to extract data used to generate new calls or to build the output returned to the user. Unlike the warehouse approach where data is transfered and stored in a database system, we choose not to materialize the view in order to insure our users to access up-to-date data.

Our approach has been developed and demonstrated as part of the multidatabase system supporting queries via uniform Object Protocol Model (OPM) interfaces.

Keywords: Information Retrieval, Information Extraction, XML, Web, Object Views

1 Introduction

The growth of the Internet has significantly changed the way information is managed and accessed. However, the success of database systems invites users to expect the best: the ability to query any source with a real rich query language such as SQL or OQL. To satisfy users, we propose an approach based on a non-materialized object view of Web datasources. Each query against the object view generates calls sent to the Web datasource. Retrieved documents are parsed, data is extracted and used to generate new calls or returned to the user.

In general, the task of wrapping data from a database system to another is a difficult task when the data model and query language of both sides are known. Web datasources are semi-structured [Abi97, Bun97] and their partially unknown structure enhances the difficulty of wrapping. Moreover, access facilities available for Web sources are rather poor. Web datasources have a few entry points (keys in a CGI form, for example) which combined with browsing allow to retrieve relevant documents. A view of a Web datasource is constrained by both the access and browsing facilities provided by the source and the extraction capabilities of the system.

In this paper, we describe the Web wrapper implemented under the OPM multidatabase system [CKMS98], based on the Object Protocol Model (OPM) [CM95] to design object views [CKMS97] of the sources and its query language, an OQL-like query language [Bea97]. The OPM Web wrapper is composed of a retrieval component and a XML engine. The former generates the calls against the Web datasource, when the latter extracts the data from cached documents. We illustrate our approach with Web biomedical datasources such as GeneCards [Gen] that integrates information about the functions of genes and their products as well as information about the biomedical applications based on this knowledge, or SwissProt [BA99], a curated sequence database which strives to provide a high level of annotations (such as the description of a protein, its domains structure, post-translational modifications, variants, etc.) a minimal level of redundancy and high level of integration with other databases. Among the characteristics of our approach we can cite the following.

Full-text search: To design an object view of a Web datasource, we first define a *search view* that express the mapping between the Web datasource and its object view. Obviously a pure object approach would dramatically restrict the expressive power of users' queries. Web datasources are mostly textual and usually provide full-text search access (search for a pattern or expression of a boolean expression of patterns) and the standard for the Object Query Language (OQL) [Bea97] does not allow the user to express full-text search queries. The mapping expressed in our search views takes advantage of all provided search facilities to allow the user to express full-text conditions against the object view.

Explicit links: Our approach is a multidatabase approach as opposed to a mediated approach. Web datasources are connected and browsing a Web datasource, one easily jumps to

another. For example, when clicking on a protein product of a genecard in GeneCards [Gen], a user sends a call to SwissProt [BA99]. Web datasources are explicitly linked when their object views are designed. This can can be done in two ways. Either each Web datasource has its own object view and links are defined at the level of the multidatabase system, or Web sources are merged in a single object view and the links are defined at the level of the search view.

Non-materialized view: The non-materialization is motivated by the need to access up-to-date data. Indeed, Web datasources such as GeneCards or SwissProt are constantly updated.

Automated browsing: Our Web wrapper successively generates as many calls as necessary to complete the search. For example a query against GeneCards using a full-text search condition returns to the wrapper a list of relevant genecards (see Figure 4). For each genecard, the wrapper extracts the name (HUGO name) and generates a call to retrieve its full description (see Figure 3).

XML format: Each retrieved document is parsed and cached in XML format. The extractor component of the Web wrapper is an XML engine. When data providers generate XML format, our XML engine will only translate the data from one DTD to another.

2 Architecture

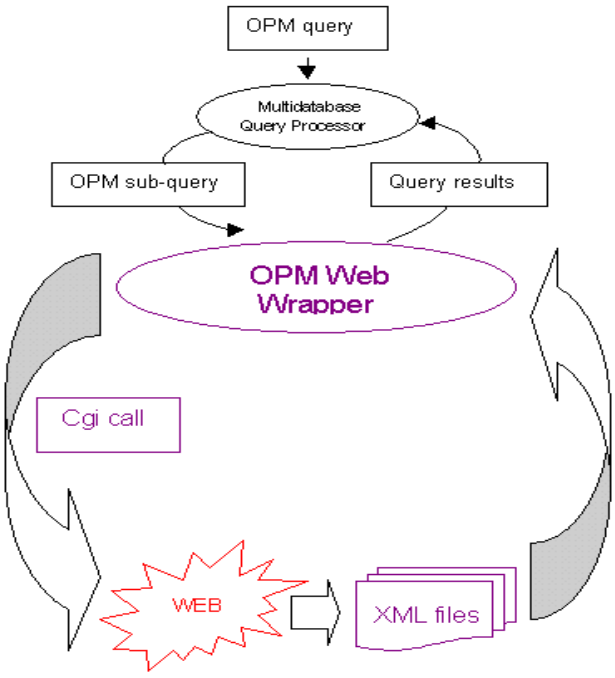

Figure 1: Architecture of the OPM Web wrapper.

As illustrated in Figure 1, our system is composed of a multidatabase system and a Web wrapper, itself composed of a retrieval and an extraction component.

We briefly present some features of the OPM Multidatabase Query System (MQS) [CKMS98] based on the OPM model [CM95]. It follows the ODMG standards [Bea97] with objects identified by a unique object identifier, qualified by attributes and classified into classes. Attributes are either simple or tuple of simple attributes and classes are organized within a sub-classing relationship. OPM*QL is an object-oriented query language similar to OQL [Bea97].

```
DATASOURCE GeneCards
 CLASS  Genecard
 WRAPPER perl wrapper.perl
 KEY ATTRIBUTE hugo_name
     CGI http://bioinformatics.weizmann.ac.il
         /cards-bin/carddisp?
     PARSER GeneCard
 SEARCH ATTRIBUTE text_search
     CGI http://bioinformatics.weizmann.ac.il
         /cards-bin/cardsearch.pl?
     PARSER SearchCard

DATASOURCE SwissProt
 CLASS  Protein
 WRAPPER perl wrapper.perl
 KEY ATTRIBUTE swiss_prot_id
     CGI http://www.expasy.ch/cgi-bin/get-spr
         ot-entry?
     PARSER SwissProt
```

Figure 2: Search view

The design of the OPM view is done together with the *search view* which expresses the mapping between the Web datasource and its object view. In a search view (see Figure 2), each class has a *key attribute* and optional *search attributes*. A key attribute is associated with a CGI call that retrieves a document containing the whole description of an instance of a class extracted with a parser. For example, the key attribute `hugo_name` of class `GeneCard` is associated with the URL `http://bioinformatics.weizmann.ac.il/cards-bin/carddisp?` and the parser `GeneCard` to parse documents such as given in Figure 3.

A search attribute is associated with a CGI call that retrieves a list of instances that satisfy a search condition expressed in the query with the `MATCH` operator. Finally, whenever a full-text search call is available, a search attribute `text_search` associated with a CGI call that returns a list of instances of any class of the datasource is defined (see Figure 4). Each class of the search view defines an object class with its key attributes and its *extracted attributes*.

Query evaluation is performed in three successive steps. First MQS processes and splits the input query in such a way that each sub-query consists in retrieving a single set of objects

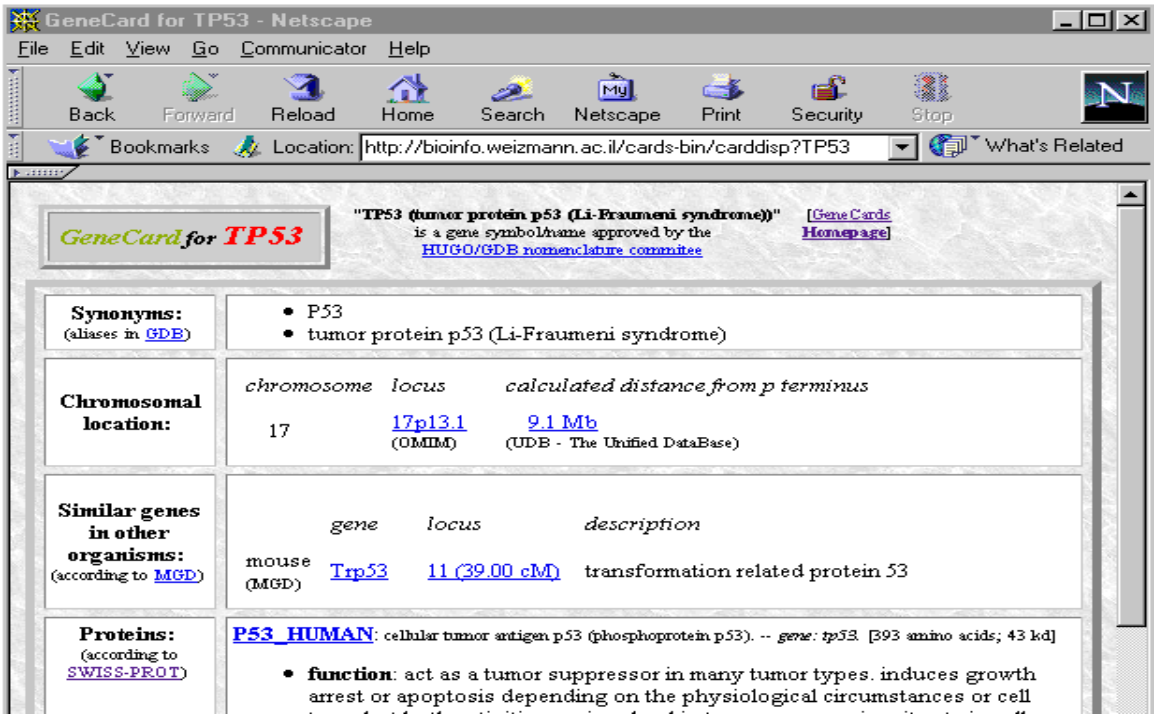

Figure 3: Genecard for TP53.

of an OPM class, and orders the generated sub-queries. Secondly, sub-queries are successively sent to the Web wrapper to retrieve documents and extract expected data. Finally, MQS evaluates the query over the data sent back by the wrapper. The retrieval component processes the query sent by MQS and generates the CGI calls reading the search view. To perform its task, it associates a *search value* to each sub-query as follows. A *search condition* is an expression of the form `@o.attribute = "string"` or `@o.attribute MATCH "string"` where `attribute` is either a key or search attribute of the class where `@o` belongs. A search condition is of search value `true` if `"string"` is a non empty string. Any other condition is of search value `false`. A conjunction of expressions has a true search value if any of its sub-expressions is of true search value. A disjunction of expressions has a true search value if both its sub-expressions are true. A negation is of search value false. The retrieval component generates one or more retrieval call(s) from queries of true search value. It also associates to each call a file name to cache the returned document. It sends the file names to be processed to the extraction component, an XML engine. As soon as it is retrieved from a Web source, a document is parsed and all the data it contains (the value of all attributes of the OPM class) is extracted and cached in XML format.

Consider the query "what are the HUGO names of the genes in GeneCards such that the strings `"TP53*"` and `"apop*"` occur in their description and their chromosomal location is on chrmosome `17`", expressed in OPM in Figure 5. The

```
SELECT g.hugo_name
FROM   g in GeneCards:Genecard
WHERE  g.text_search MATCH "TP53* AND apop*"
  AND  g.chromosomal_loc.chromosome = "17"
```

Figure 5: OPM query

search value of the query is true and the CGI call `http://bioinformatics.weizmann.ac.il/cards-bin/cardsearch.pl?TP53*+AND+apop*` is generated and sent to the Web datasource. The output (see Figure 4) is retrieved and cached in XML format as illustrated in Figure 6.

```
<?xml version="1.0"?>
<GeneCards>
<Genecard oid="TP53">
  <text_search>TP53* AND apop*</text_search>
  <hugo_name>TP53</hugo_name>
</Genecard>
<Genecard oid="TNFRSF10A">
  <text_search>TP53* AND apop*</text_search>
  <hugo_name>TNFRSF10A</hugo_name>
</Genecard>
   ...
</GeneCards>
```

Figure 6: Output cached in XML format

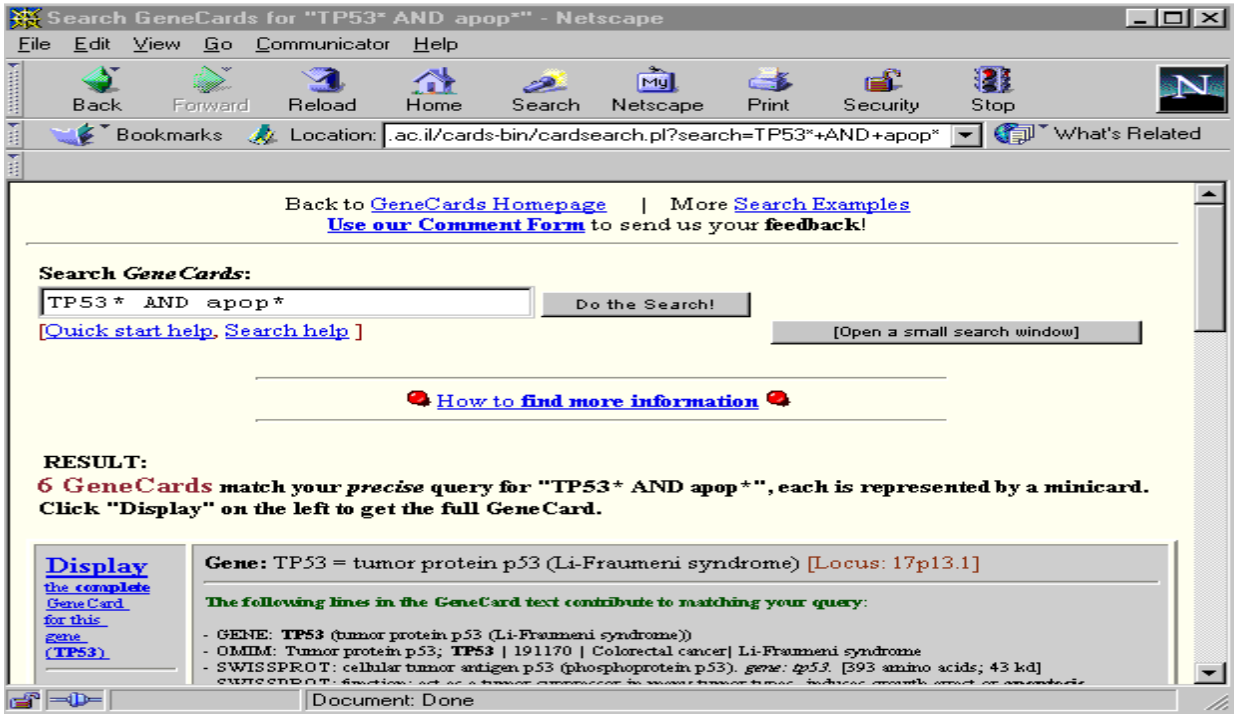

Figure 4: Full text search for TP53* AND APOP*.

Given a sub-query, the XML engine generates an output template, fills it up with data extracted from the cached documents and returns the output to MQS. Whenever data is missing to return the output to MQS, the XML engine generates the queries corresponding to the missing data and sends them to the retrieval component for more CGI calls until the output is complete and is returned to MQS. In particular, the latter happens when a call is generated from a search condition built with a search attribute. To answer the query given in Figure 5, the value of attribute `chromosome_loc` is missing. In search mode, the XML engine extracts the HUGO name of the six genecards satisfying the search condition to generate six CGI calls and retrieve the full description of each genecard. It returns the value of the attributes `text_search`, `hugo_name` and `chromosomal_loc` of the six genecards to MQS for evaluation. The result of the query is `TP53` since only this gene is located on chromosome 17.

3 Conclusion

In this paper we describe an approach to access Web datasources through an object database view. The object schema corresponds to the users' understanding of the data and the query language facilitates the expression of queries with path expressions. The Web wrapper is generic and can be easily extended to new Web sources given their CGI accesses and corresponding parsers. Each Web datasource is described in a *search view* that characterizes the mapping between each Web datasource and its object view. Web datasources are usually flat file sources and are associated with a search engine to retrieve relevant documents. Our object views allow the user to express search conditions when asking queries to take full advantage of the available search facilities provided by the sources.

The use of a multidatabase system considerably extends the approach with the ability to combine data retrieved from the Web with data extracted from other sources. Not only these datasources can be queried with a real query language but the user is able to ask queries over several sources explicitly linked in a global OPM schema. The XML engine is also used as the extraction component for local textual sources stored in XML [Lac99].

The OPM Web wrapper has been implemented by the Data Logic group at Gene Logic. The OPM Web wrapper is currently able to process textual information. However, Web biomedical datasources also provide pictures and maps. An extension of our approach to map methods specific to display images available at the multidatabase system level [TKM99] to the corresponding call to retrieve the data on the Web source is currently under implementation.

Acknowledgments: The authors wish to thank the Data Logic group at Gene Logic for their useful comments when designing and implementing the wrapper for Web sources. In particular Anthony Kosky is thanked for answering all questions regarding the OPM multidatabase system and its implementation.

REFERENCES

Abi97. S. Abiteboul. Querying semi-structured data. In *Proc. of Intl. Conf. on Database Theory*, pages 1–18, Delphi, Greece, January 1997. LNCS 1186, Springer.

BA99. A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence databank and its supplement TrEMBL. *Nucleic Acids Res*, 1(27):49–54, January 1999. http://www.expasy.ch/sprot.

Bea97. D. Bartels and et al. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, San Francisco, 1997.

Bun97. P. Buneman. Semistructured data. In *Proc. ACM Symp. on Principles of Database Systems*, Tucson, 1997. Invited tutorial.

CKMS97. I.A. Chen, A.S. Kosky, V.M. Markowitz, and E. Szeto. Constructing and Maintaining Scientific Database Views. In *Proceedings of the 9th Conference on Scientific and Statistical Database Management*, August 1997.

CKMS98. I.A. Chen, A.S. Kosky, V.M. Markowitz, and E. Szeto. Exploring Heterogeneous Biological Databases: Tools and Applications. In *Proceedings of the 6th Conference on Extending Database Technology*, March 1998. http://www.genelogic.com/opm.htm.

CM95. I.A. Chen and V.M. Markowitz. An Overview of the Object-Protocol Model (OPM) and OPM Data Management Tools. *Information Systems*, 20(5):pp 393–418, 1995.

Gen. GeneCards. http://bioinformatics.weizmann.ac.il/cards/. Weizmann Institute Genome Center and Bioinformatics Unit.

Lac99. Z. Lacroix. A XML Engine to query Flat File and Web datasources through Object Database Views. Technical report, March 1999.

TKM99. T. Topaloglou, A. Kosky, and V. Markovitz. Seamless Integration of Biological Applications within a Database Framework. Technical report, 1999.