Biological Data Integration: Wrapping Data and Tools

Zoé Lacroix

***Abstract*—Nowadays scientific data is inevitably digital and stored in a wide variety of formats in heterogeneous systems. Scientists need to access an integrated view of remote or local heterogeneous data sources with advanced data accessing, analyzing, and visualization tools. Building a digital library for scientific data requires accessing and manipulating data extracted from flat files or databases, documents retrieved from the Web as well as data generated by software. We present an approach to wrapping web data sources, databases, flat files, or data generated by tools through a database view mechanism. Generally, a wrapper has two tasks: it first sends a query to the source to retrieve data and, second builds the expected output with respect to the virtual structure. Our wrappers are composed of a retrieval component based on an intermediate object view mechanism called *search views* mapping the source capabilities to attributes, and an eXtensible Markup Language (XML) engine, respectively, to perform these two tasks. The originality of the approach consists of: 1) a generic view mechanism to access seamlessly data sources with limited capabilities and 2) the ability to wrap data sources as well as the useful specific tools they may provide. Our approach has been developed and demonstrated as part of the multidatabase system supporting queries via uniform object protocol model (OPM) interfaces.**

***Index Terms*—Biological data integration, database view, eXtensible Markup Language (XML), mediation, web data sources.**

I. Introduction

A LARGE part of the work of scientists today consists in querying multiple remote or local heterogeneous data sources, integrating manually retrieved data and manipulating it with advanced data analyzing and visualization tools. In each scientific domain, the access to the relevant data, combining data sources and coping with their distribution and heterogeneity is a tremendously difficult task.

An important aspect of bioinformatics consists in building a scientific digital library, integrated view of all data of interest widely distributed and constantly updated in heterogeneous remote public data sources or local private ones. Biological data is available in a wide variety of formats, annotated, and stored in flat files and relational or object-oriented databases. Access to heterogeneous biological data sources is mandatory to scientists. A single query may involve flat files such as GenBank [10] or SwissProt [7], web resources such as LENS (Linking Expressed Sequence Tags and their associated Name Space) [39], the Saccharomyces Genome Database [43], or the references data source PubMed [42]. These sources are textual and of restricted access facilities. Their structure varies from ASN.1 data exchange format for GenBank to poorly structured HTML format for LENS. Biological data sources do not provide users with a real expressive query language to access the data they contain. Instead, they provide a wide range of useful tools such as text search engines or similarity search tools such as FASTA [41], BLAST [3], or LASSAP [30] that generate additional information needed to access the data.

The database community has devoted a large amount of work on integration of data either materialized within data warehouses or nonmaterialized with a middleware system. Data integration consists in wrapping data sources and either loading retrieved data into a data warehouse or returning it to the user. Wrapping a data source means retrieving data from the source and translating it into the common integrated data representation. Some systems have been designed for the specific integration of biomolecular data. Among them, one can cite K2 [23], Pizzkell/Kleisli [47], the multidatabase system based on the object protocol model (OPM) [21], DiscoveryLink [34], extension for life science of the DataJoiner based upon DB2 [17] merged with Garlic [18], P/FDM [35], or TAMBIS [8]. K2, Pizzkell/Kleisli, and TAMBIS are built upon Kleisli [9], [25] using its complex data model and query language CPL [12]. K2 and Kleisli rely on a set of data drivers to handle as many data sources. K2 provides a new language K2MDL, whereas additional programming can be used in Kleisli [46], [47] with its Pizzkell suite of interfaces to capture more complex data access such as retrieval facilities provided by Web data servers. TAMBIS provides a transparent integrated view of data sources through a pool of concepts that capture predefined data accesses and thus limited queries and offers no link to information retrieval tools and requires the implementation of a new wrapper for each new data source. P/FDM also requires additional coding to access specific sources capabilities such as SRS [26]. DiscoveryLink encapsulates the access to specialized search capabilities into user-defined functions. The OPM multidatabase system is based on the OPM [22] to design object views [20] of the sources, and its query language, an OQL-like query language. It has a convenient architecture and APIs for its extension to new wrappers, as CORBA servers. However, tools such as BLAST are wrapped through a specifically designed OPM class called application specific data type (ASDT) [45] that requires the implementation of a corresponding CORBA server to wrap the tool with respect to this upper level OPM representation. None of these approaches provides a seamless integration that permits the use of metadata about source content and access cost to complex source capabilities such as sequence similarity search into the query planning phase and, thus, permits the optimization of the evaluation of the queries.

The whole problem of designing a virtual database schema is to provide the user with the best user's view of the data sources when the access to the data is constrained by the source *retrieval*

Manuscript received December 4, 2000; revised December 6, 2001.

Z. Lacroix is with the Arizona State University, Tempe, AZ 85287-6106 USA (e-mail: zoe.lacroix@asu.edu).

Publisher Item Identifier S 1089-7771(02)04900-2.

and *extraction* facilities. The retrieval step consists in selecting documents out of (a large number of) documents, where a document is a formatted file (flat file, output of a database query or a tool, or an HTML output). Retrieval may be done through specific limited accesses (source identifiers, for example), an indexing system such as SRS [26], a search engine, a similarity search on sequences, or any other tool. Extraction consists in parsing the retrieved documents and identifying the data they contain for query evaluation. Clearly, the source capabilities restrict the ability to access data and provide a limited data representation regardless of the richness of the source data schema. All the above cited approaches focus on providing an integrated interface to biologists through a relational, object-oriented, or conceptualized views of data sources based on a *global as view* approach. A global as view approach consists in designing the user's data organization as a view of all data sources. It makes the query evaluation easy by translating straightforwardly the user's query into source queries. However, each wrapping of a data source requires some work and constant changes made by data providers make it difficult to maintain wrappers. Conversely, a *local as view* approach expresses each data source as a view of the integrated user's organization of data (see [40] for a local-as-view-based system). This approach makes the maintenance of wrappers easier but to the price of a costly and difficult query evaluation. In addition, both view mechanisms are relying on a pure database approach and, thus, no tool or search facility is made available to the user. Data retrieval through limited capabilities justifies a local as view approach with the source capabilities as a view of the user's data representation. However, the documents returned by the sources offer a much richer data representation that can be used as long as the data can be extracted. The latter motivates a global as view approach upon semistructured documents (the schema is not always known).

We propose to combine local and global as view approaches to access heterogeneous data sources and their available tools based on an intermediate object view mechanism called *search views* [36]–[38]. Search views express the mapping between the object view and the access capabilities available to retrieve the data from web sources. Search views can map *direct calls* retrieving a document that completely describes an object, *search calls* to the search engines retrieving a document containing a list of objects, and *abstract calls* corresponding to hyperlinks between objects or tools generating objects. The extraction component processes the retrieved files, extracts the data, and returns it in the user's format for query evaluation. The emergence of the eXtensible Markup Language (XML) as a new standard for data representation and exchange on the web motivates the choice of XML [13] as an intermediate format. All retrieved documents are parsed and cached in XML format. When the design and implementation of our approach was started, no data provider generated documents in XML and retrieved documents needed to be translated into XML for caching. Today, many genomic data providers deliver data in XML format.

In Section II, we describe the search views that represent the resources capabilities. The general architecture and and the query processing are presented in Section III. In Section IV, we conclude and discuss future extensions.

II. Search Views

In general, the task of wrapping data from a database system to another is a difficult task, but at least the data model and query language of both sides are known. Flat files and web sources do not have a clear data model, but are semistructured [1], [14]. If flat file data sources can provide a language to retrieve data thanks to a sophisticated indexing system, the search facilities available for web sources are rather poor. Each Web data source has entry points (keys in a CGI form, for example), which allow to retrieve documents either describing completely an object (*direct calls*) or providing a list of relevant objects (*search calls*). A retrieval query must correspond to one of these entry points. The retrieval component has the hard task to map a real database query with CGI calls. To accomplish its tasks, the retrieval component of the object web wrapper (OWW) presented in this paper uses a view mechanism based upon *search views* introduced in [36]. Search views list all data sources capabilities through *attributes* and the corresponding document processing tools used to parse retrieved documents and cache them in XML for data extraction.

A. Retrieval Capabilities

In a search view, each data source is represented by its name and the list of its classes. Direct calls are represented as *key attributes*,[1] when search calls are associated with *search attributes* [37]. Most web data sources only provide data embedded into an HTML shell for display. However, there are sources where HTML, and in particular its hyperlinks, is used to structure the data. Hyperlinks can be often represented in the object-oriented database framework as *isa* relationships or as abstract attributes. In addition, tools such as similarity search can also be represented as abstract attributes. The search view and the associated view mechanism presented in [36] and [37] are extended to take advantage of abstract attributes [38]. Each class must present at least a key attribute in addition to as many search or abstract attributes as it is possible to define.

An *abstract attribute* in a search view links an object to one or several objects of a target data source and class. It can be either single valued or set valued. A single-valued abstract attribute is associated with a direct call and the attribute whose value is used to build the call. A set-valued attribute is associated with a search call and the attribute(s) whose value(s) is (are) used to build the call.

Consider PubMed [42], the National Library of Medicine's search service that provides access to over 10 000 000 citations. It provides direct access to citations given a PubMed or a MEDLINE identifiers. It also enables typed search calls (through author names, journal titles, etc.) as well as untyped search call (fulltext index through the whole description of the citation). Given a citation, it also provides links to similar documents and NCBI databases. The latter link can be represented as an abstract attribute. The user's object view of PubMed defines an abstract attribute `similar_articles` linking the object class `Citation` to itself (because retrieved objects are instances of class `Citation`). In the search view of PubMed (see Fig. 1), `similar_articles` is represented as an abstract attribute, asso-

[1]Key attributes were called *search attributes* in [36].

```
DATA SOURCE PUBMED
 CLASS  Citation
 KEY ATTRIBUTE pubmed_id
   CALL http://www.ncbi.nlm.nih.gov/cgi-bin/
       Entrez/referer?/htbin-post/
       Entrez/query%3fdb=m&form=6&uid=
   PARSER PubMed
 KEY ATTRIBUTE medline_id
   CALL http://www.ncbi.nlm.nih.gov/cgi-bin/
       Entrez/referer?/htbin-post/
       Entrez/query%3fdb=m&form=6&uid=
   PARSER PubMed
 SEARCH ATTRIBUTE text_search
   CALL http://www.ncbi.nlm.nih.gov/htbin-post/
       Entrez/query?form=4&db=m&term=
   PARSER PubSearch
 ABSTRACT ATTRIBUTE similar_articles
   EXTRACTED PARAMETER pubmed_id $pubmed_id
   TARGET DATA SOURCE PUBMED
   TARGET CLASS Citation
   CALL http://www.ncbi.nlm.nih.gov/htbin-post/
       Entrez/query?db=m&form=6&uid=$pubmed_id
   PARSER PubSearch

DATA SOURCE GENBANK
 CLASS Sequence
 KEY ATTRIBUTE genbank_id
   CALL http://www.ncbi.nlm.nih.gov:80/entrez/
       query.fcgi?cmd=Retrieve&db=Nucleotide&
       list_uids=7184999&dopt=GenBank
   PARSER GenBank
 ABSTRACT ATTRIBUTE blast_call
   EXTRACTED PARAMETER query $query
   GIVEN PARAMETER program $program
         DEFAULT blastn
   GIVEN PARAMETER database $database
         DEFAULT swissprot
   GIVEN PARAMETER ungapped_alignment $ung_al
         DEFAULT is_set
   GIVEN PARAMETER pairwise_alignement $al
         DEFAULT 0
   GIVEN PARAMETER query_anchored_w_id $al
   GIVEN PARAMETER query_anchored_wo_id $al
   GIVEN PARAMETER description $description
         DEFAULT 100
   GIVEN PARAMETER alignments $alignments
         DEFAULT 50
   TARGET DATA SOURCE BLAST
   TARGET CLASS Sequence
   CALL http://www.ncbi.nlm.nih.gov/blast/Blast
   PARSER BlastHits
```

Fig. 1. Search view.

ciated to a search call (the abstract attribute is set-valued) that requests a PubMed identifier (extracted attribute) as input to retrieve the citations (target class) of PubMed (target data source). Similarly, abstract attributes can be used to represent tools available at the web data source.

As of October 2000, GenBank [10] provides access to its 10 336 000 000 bases in 9 103 000 sequences [31] through a variety of typed and untyped search facilities with the Entrez interface and BLAST similarity search tools. The access provided by Entrez can be expressed as a `text_search` or search attributes, when the access to BLAST can be expressed by an abstract attribute. A BLAST call is built from a `query` (either the extracted attribute `sequence` from the description of a retrieved object of class `Sequence` in GenBank, or a sequence provided by the user in his query) and several parameters such as the BLAST `program` name (if the user does not provide the value for these parameters, default values are given in the search view).

B. Document Processing

Retrieved documents must be processed to extract the data relevant to answer the query. There is a need for data transformation rather than data extraction only. Indeed, most of these data sources do not represent objects in a way that facilitates data integration and data querying. For example [27], GenBank and SwissProt are both sequence-centric when a user would expect them to be, respectively, gene-centric and domain-centric. As a consequence, the data structure organizing retrieved data does not match the user's object view customizing his needs in terms of integration and querying.

In a search view, each attribute is associated to a parser that processes retrieved documents and caches them in the XML format matching the OPM view. Each OPM class name corresponds to an XML element definition and each attribute of the class corresponds to a an XML subelement. The parsers perform three tasks: 1) they extract data from retrieved documents; 2) restructure the data with respect to the user's view; and 3) cache them in XML.

When this approach has been designed and implemented, no data provider delivered data in XML and we had to adopt an alternate plan to accomodate HTML. Our experience had proven that the output of web data sources often is not in "pure" HTML format, and parsers designed for pure HTML documents usually fail. Moreover, HTML tags do not structure the output: the data is mostly textual and often only embedded into an HTML shell for display. For all these reasons, the parsers we used in our alternative plan to process HTML documents retrieved from the Web were manually generated PERL subroutines. We decided not to use any automatic wrapper generator (see, for instance, [4], [16], [19], and [33]) because generated wrappers would only accomodate pure HTML and would not extract data structured by ASCII tabulations or other pure textual formating. The additional work needed to extend these generated parsers appeared to be more costly than to write them by hand from scratch. In addition, automatic wrapper generation approaches do not provide maintenance flexibility in the sense that when the format of the data source has changed slightly, one still has to generate a new wrapper from scratch. Any additional work that had to be performed to adapt the previously generated wrapper to extract all data of relevance would have to be done from scratch again.

The advantage of our approach is that whenever a data provider publishes data in XML format the search view can be easily updated by replacing the PERL parsers by XML queries. For example, the BLAST call defined at GenBank in Fig. 1 can associated to an XML query `BlastHits` that processes all retrieved hits and cache them. (BLAST output provided by GenBank is available in a variety of formats including XML.) XML queries [15] are expressions that, given an input in XML format, extract the data, manipulate it, and return it in XML format (the output XML schema [28], [44], [11] may be different than the input XML schema). For the sake of compatibility and reusability we choose to use expressions of a language that have been designed to be a standard rather than similar approaches such as XML-QL [24], Strudel [29] or Lorel [5], [32]. It is worth noting that our approach can process documents generated in HTML and documents generated in XML since each data format is processed by a specific associated parser. PERL subroutines process HTML documents and XML queries process XML documents. Both types of parsers cache the processed documents in XML. A single XML engine performs final XML extraction to return the data to the multidatabase system.

III. Architecture

The OWW was designed to perform as a component of the OPM multidatabase query system based on the OPM model [22]. It follows the ODMG standards [6] 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 subclassing relationship. OPM*QL is an object-oriented query language similar to the object query language (OQL) [6]. We do not discuss any of the multidatabase aspects of the system in the paper and invite the reader to read [21] for a complete description.

A. Design of a View of a Web Data Source

The design of the OPM schema of a web data source is restricted to the retrieval and extraction capabilities of the wrapper. Retrieval capabilities are expressed in the *search view* as key, search or abstract attributes associated to classes. Only objects of the classes defined in the search views can be accessed directly from a user's query. Additional classes can be defined in the OPM schema but their objects can only be accessed through objects of search classes. Each search class defines an OPM class. All attributes mentioned in the search view must be defined at their corresponding OPM class. The structure of retrieved Web documents is usually richer than the attributes expressed as retrieval capabilities (through indexes). All extractable attributes correspond to XML elements (tags) in cached documents and are also defined at their corresponding OPM class. Therefore, the search view expresses the retrieval capabilities as a local-as-view approach, when the OPM view is the extension of the search view to all extraction capabilities, as a global-as-view approach.

To extend our approach to a new data source, one has to perform the following:

1) identify the accesses (direct, search, or abstract) available at the web data source;
2) design the search view corresponding to these accesses;
3) write parsers (HTML or XML queries);
4) register the parsers in the search view;
5) design the object (user's) view by collecting all information defined at the search view and extending it to all extractable information.

The maintenance of the approach consists in replacing the parsers when the format of retrieved documents has changed, replacing the calls when the URL has changed, adding or removing attributes in the search view when the accesses to the data source have changed, and forwarding these changes to the object view when applicable.

B. An Example

Suppose that a biologist wishes to retrieve all abstracts of papers mentioning a given gene, all sequences mentioned in each paper and all accession numbers corresponding to similar sequences. To evaluate this query, a user can use a browser and accesses PubMed, fills up the form with the gene name, retrieves all PubMed citations and extracts for each of them the GenBank accession numbers, accesses GenBank, fills up the form with the accession number, retrieves each sequence description, and extracts the sequence.

Using our approach instead automatizes entirely the process and the user only has to type the following query (or generate it through a form-based interface):

```
select  @c.abstract,
        @s.sequence,
        @b.accession
from    @c in PubMed:Citation,
        @s in GenBank:Sequence,
        @b in Blast:Sequence
where   @c.text_search = "P53"
  and   @c.genbank_id = @s.accession
  and   @s.blast_call = @b
  and   @s.sequence = @s.sequence
  and   @b.program = "blastn"
```

To incorporate PubMed and GenBank in the system, one designs a search view such as given in Fig. 1 and an OPM view extended to all extractable attributes.[2]

The multidatabase system forward the query directly to the OWW since it only involves Web data sources. The OWW processes the query as follows: 1) it generates a retrieval query to PubMed with the string "P53"; 2) the XML engine extracts all GenBank identifiers and abstracts from retrieved documents; 3) it generates new calls to retrieve all sequences descriptions from GenBank; 4) it extracts sequences from documents retrieved from GenBank; 5) it sends as many BLAST calls to retrieve similar sequences; and 6) it returns abstract sequences and the accession numbers of similar sequences to the multidatabase system. The multidatabase system evaluates the query against data returned by the OWW and returns the result to the user.

C. Query Processing

The OWW as described in the paper has the following tasks to perform: 1) given an OQL query, it generates the retrieval queries needed to be sent to each web data source to retrieve all the data necessary to evaluate the query and 2) it extracts the data from retrieved documents and returns it to the database sytem for evaluation and visualization. The OWW has two components: a *retrieval component* built upon the intermediate search views of web data sources that generates the retrieval

[2]Both data sources provide multiple accesses of interest to the user; the ones mentioned in Section II and defined in Fig. 1 only constitute a small subset of them.

queries and an *XML extractor* that processes retrieved documents, extract the data and returns them to the database system. However, in addition to the requirements specified above, the retrieval component of the OWW also achieves 3) partial evaluation and 4) optimization in the query planning phase.

Given a query, the retrieval component generates a query plan based on search conditions extracted from the `WHERE` clause of the query and associated with search values. Selected search conditions generate retrieval queries (calls) to the Web data source. The query planning described in details in [37] is extended to generate retrieval queries corresponding to abstract attributes in a second step. When all retrieval queries corresponding to key and/or search attributes are generated and documents are processed and cached in XML, new retrieval queries are generated. Abstract attributes require data extracted from retrieved document to build their associated call. When the data are extracted by the XML engine, the retrieval component can submit new calls corresponding to the abstract attributes. The loop retrieval/extraction ends when all retrieval queries generated by the plan can be submitted.

Before sending any call to the web source, the retrieval component checks whether or not the documents are already cached. Indeed, during a query session, a complicated query (involving many joints on many data sources) usually requires data extracted from the same documents at different timepoints in the query evaluation. To optimize the approach by avoiding sending multiple identical calls, the retrieval component of OWW performs cache checking prior to submiting a call. Had the document been already cached or had it been retrieved by a new call, the retrieval component of the OWW assigns names to the cached files and sends the file name(s) back to the XML engine. The current implementation assumes that, for the query to be completely evaluated, there are enough search conditions in the query to generate retrieval queries and all retrieval queries success.

IV. Conclusion

Querying scientific data and, in particular, biomolecular data, requires the integration of widely distributed heterogeneous data sources and advanced data analysis and visualization tools. A large amount of scientific data is semistructured and stored in flat-files or published on the Web and must be integrated with structured data available in relational or object-oriented databases.

The OWW presented is this paper is based on the notion of *Search View* presented in [36] extended to access most of the source capabilities. The design of a search view of a web data source only requires to know what are the available calls or parameterized URLs to access the search engine and to write XML queries to translate retrieved documents into the XML schema corresponding to the user's view. As long as the source capabilities are expressible with search views, no additional programming effort is required to access any new web data source thanks to the OWW search view mechanism. Our approach allows typed search (with wildcards on attributes) or untyped search (with the `text_search` attribute), but it could be further extended to allow a user to search over the object structure of the view like in OQL-doc [2] with generalized path expressions. Such an extension would enable the user to express queries without a full explicit knowledge of the object structure.

Biologists aim to submit queries involving multiple joints over several data sources and invoking many tools. These queries are not processed efficiently by the OWW because only basic query optimization is performed. We forsee the need for developing optimization techniques at the query planning phase in order to evaluate queries more efficiently. An approach such as the OWW based on search views provides the necessary framework for advanced query planning and optimization.

Acknowledgment

The author wishes to thank the Data Logic Group at Gene Logic, Inc. for making the implementation of this work possible. They would also like to thank B. Eckman for motivating this work and contributing helpful comments.

References

[1] S. Abiteboul, “Querying semi-structured data,” in *Proc. Int. Conf. Database Theory*, Delphi, Greece, Jan. 1997, pp. 1–18. LNCS 1186.
[2] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, and J. Siméon, “Querying documents in object databases,” *J. Digital Libraries*, 1997.
[3] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. (1990, Oct.) Basic local alignment search tool. *J. Molecular Biol.* [Online], vol (3), pp. 403–410. Available: http://www.ncbi.nlm.nih.gov/BLAST
[4] N. Ashish and C. Knoblock, “Wrapper generation for semi-structured internet sources,” presented at the ACM SIGMOD Workshop on Management of Semistructured Data, Tucson, AZ, May 1997.
[5] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. (1997) The lorel query language for semistructured data. *J. Digital Libraries* [Online]. Available: ftp://db.stanford.edu/pub/papers/lorel96.ps
[6] D. Bartels *et al.*, *The Object Database Standard: ODMG 2.0*. San Francisco, CA: Morgan Kaufmann, 1997.
[7] A. Bairoch and R. Apweiler. (1999, Jan.) The SWISS-PROT protein sequence databank and its supplement TrEMBL. *Nucleic Acids Res.* [Online], vol (27), pp. 49–54. Available: http://www.expasy.ch/sprot
[8] P. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens, “TAMBIS: Transparent access to multiple bioinformatics information sources. An overview,” presented at the Proc. 6th Int. Conf. Intell. Syst. Molecular Biology (ISBM98), 1998.
[9] P. Buneman, J. Crabtree, S. Davidson, V. Tannen, and L. Wong, *BioKleisli: Integrating Biomedical Data and Analysis Packages*. Boston, MA: Kluwer, 1999, ch. 17.
[10] D. Benson, I. Karsch-Mizrachi, D. Lipman, J. Ostell, B. Rapp, and D. Wheeler. (2000, Jan.) GenBank. *Nucleic Acids Res.* [Online], vol (28), pp. 15–18. Available: http://www.ncbi.nlm.nih.gov/Genbank
[11] XML Schema Part 2: Datatypes, P. Biron and A. Malhotra. (2001). [Online]. Available: http://www.w3.org/TR/2001/REC-xmlschema-2-20 010 502
[12] P. Buneman, S. Naqvi, V. Tannen, and L. Wong, “Principles of programming with complex objects and collection types,” *Theoretical Comput. Sci.*, vol. 149, no. 1, pp. 3–48, 1995.
[13] Extensible Markup Language (XML) 1.0, T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. (2000). [Online]. Available: http://www.w3.org/TR/2000/REC-xml-20 001 006
[14] P. Buneman, “Semistructured Data,” presented at the Proc. ACM Symp. on Principles of Database Systems, 1997. Invited tutorial.
[15] XQuery: A Query Language for XML, D. Chamberlin, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. (2000). [Online]. Available: http://www.w3.org/TR/xmlquery
[16] T. Critchlow, M. Ganesh, and R. Musick, “Automatic generation of warehouse mediators using an ontology engine,” presented at the Proc. 5th KRDB Workshop, Seattle, WA, 1998.
[17] D. Chamberlin, *A Complete Guide to DB2 Universal Database*. San Francisco, CA: Morgan Kaufmann, 1998.

[18] W. F. Cody, L. M. Haas, W. Niblack, M. Arya, M. J. Carey, R. Fagin, D. Lee, D. Petkovic, P. M. Schwarz, J. Thomas, M. Tork Roth, J. H. Williams, and E. L. Wimmers. Querying multimedia data from multiple repositories by content: The garlic project. presented at Proc. IFIP 2.6 3rd Working Conf. Visual Database Syst. (VDB-3). [Online]. Available: http://www.almaden.ibm.com/cs/garlic

[19] T. Catarci, L. Iocchi, D. Nardi, and G. Santucci, "Conceptual views over the web," presented at the Proc. 4th KRDB Workshop, Athens, Greece, 1997.

[20] I. A. Chen, A. S. Kosky, V. M. Markowitz, and E. Szeto, "Constructing and maintaining scientific database views," presented at the Proc. 9th Conf. Sci. Statist. Database Management, Aug. 1997.

[21] ——, "Exploring heterogeneous biological databases: Tools and applications," presented at the Proc. 6th Conf. Extending Database Technol., Mar. 1998.

[22] I. A. Chen and V. M. Markowitz, "An overview of the object-protocol model (OPM) and OPM data management tools," *Inform. Syst.*, vol. 20, no. 5, pp. 393–418, 1995.

[23] S. Davidson, J. Cabtree, B. Brunk, J. Schug, V. Tannen, C. Overton, and C. Stoeckert, "K2/Kleisli and GUS: Experiments in integrated access to genomic data sources," *IBM Syst. J.*, vol. 40, no. 2, pp. 512–531, 2001.

[24] XML-QL: A Query Language for XML, A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. (1998). [Online]. Available: http://www.w3.org/TR/NOTE-xml-ql/

[25] S. Davidson, C. Overton, V. Tannen, and L. Wong, "BioKleisli: A digital library for biomedical researchers," *J. Digital Libraries*, 1997.

[26] T. Etzold and P. Argos. (1993) SRS, an indexing and retrieval tool for flat file data libraries. *Comput. Applicat. Biosci.* [Online], vol (1), pp. 49–57. Available: http://srs.ebi.ac.uk

[27] B. Eckman, A. Kosky, and L. Laroco, "Extending traditional query-based integration approaches for functional characterization of post-genomic data," *Bioinformatics*, vol. 17, no. 7, pp. 587–601, 2001.

[28] XML Schema Part 0: Primer, D. Fallside. (2001). [Online]. Available: http://www.w3.org/TR/2001/REC-xmlschema-0-20 010 502/

[29] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu, "STRUDEL: A web-site management system," presented at the ACM SIGMOD—Research Prototype Demonstration, Tucson, Arizona, May 1997.

[30] E. Glemet and J.-J. Codani, "LASSAP: A Large scale sequence comparison package," *Bioinformatics*, vol. 13, no. 2, pp. 137–143, 1997.

[31] Growth of Genbank [Online]. Available: http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

[32] R. Goldman, J. McHugh, and J. Widom. From semistructured data to XML: Migrating the lore data model and query language. presented at ACM SIGMOD Workshop on the Web and Databases (WebDB'99). [Online]. Available: http://www-db.stanford.edu/lore

[33] J.-R. Gruser, L. Raschid, M. Vidal, and L. Bright, "Wrapper generation for web accessible data sources," presented at the CoopIS, 1998.

[34] L. Hass, P. Kodali, J. Rice, P. Schwarz, and W. Swope, "Integrating life sciences data—With a little garlic," presented at the Proc. IEEE Int. Symp. Bio-Informatics Biomed. Eng. (BIBE), Washington, DC, Nov. 2000.

[35] G. Kemp, N. Angelopoulos, and P. Gray, "A schema-based approach to building a bioinformatics database federation," presented at the Proc. IEEE Int. Symp. Bio-Informatics Biomed. Eng. (BIBE), Washington, DC, Nov. 2000.

[36] Z. Lacroix, "Object views through search views of web datasources," presented at the Proc. Int. Conf. Conceptual Modeling (ER99), Paris, France, Nov. 1999.

[37] ——, "Querying annotated scientific data combining object-oriented view and information retrieval," presented at the Proc. 6th Int. Conf. Content-Based Multimedia Inform. Access (Recherche d'Informations Assistée par Ordinateur) RIAO, Paris, France, Apr. 2000.

[38] ——, "Scientific data integration: Wrapping textual documents with a database view mechanism and an XML engine," presented at the Proc. IEEE Int. Symp. Bio-Informatics Biomed. Eng. (BIBE), Washington, DC, Nov. 2000.

[39] LENS [Online]. Available: http://www.agave.humgen.upenn.edu/lens

[40] A. Levy, A. Rajaraman, and J. Ordille, "Querying heterogeneous information sources using source descriptions," presented at the Proc. Int. Conf. Very Large Data Bases, 1996.

[41] W. Pearson and D. Lipman, "Improved Tools for Biological Sequence Comparison," in *Proc. Nat. Academy Sci. USA*, vol. 85, Apr. 1988, pp. 2444–2448.

[42] PubMed [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/

[43] SGD [Online]. Available: http://genome-www.stanford.edu/saccharomyces/

[44] XML Schema Part 1: Structures, H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. (2001). [Online]. Available: http://www.w3.org/TR/2001/REC-xmlschema-1-20 010 502/

[45] T. Topaloglou, A. Kosky, and V. Markovitz, "Seamless integration of biological applications within a database framework," in *Proc. 7th Int. Conf. Intell. Syst. Molecular Biology (ISBM)*, Heidelberg, Germany, 1999, pp. 272–281.

[46] L. Wong, "Some MEDLINE queries powered by Kleisli," presented at the ACCESS, June 1998.

[47] ——, "Kleisli, its exchange format, supporting tools, and an application protein interaction extraction," presented at the Proc. IEEE Int. Symp. Bio-Informatics Biomed. Eng. (BIBE), Washington, DC, Nov. 2000.

Zoé Lacroix received the Ph.D. degree in computer science from the University of Paris XI, France, in 1996.

She has been a Researcher at the French Institut National de la Recherche en Informatique et Automatique (INRIA) and at the Institute for Research in Cognitive Science (IRCS) at the University of Pennsylvania (USA). Since then she has been working at Gene Logic and at SurroMed, two biotech companies, where her research focused on bioinformatics. In addition, she has been involved in the XML working groups XML Query Language and XML Forms at the World Wide Web Consortium (W3C). Recently, she joined Arizona State University where she started new projects on electronic business hubs, integration of biological databases, optimization, semantics of Web queries, and semantic web.