
JXP4BIGI: a generalized, Java XML-based approach for biological information gathering and integration

Yihua Huang^{1,2,4}, Tianyun Ni^{1,2}, Lei Zhou^{2,3} and Stanley Su¹

¹Database Systems R&D Center, College of Engineering, ²Shands Cancer Center ³Department of Molecular Genetics and Microbiology, College of Medicine, University of Florida, FL, USA; and

⁴Department of Computer Sci. & Tech., Nanjing University, Nanjing, P.R.China.

ABSTRACT

Motivation: In the post-genomic era, biologists interested in systems biology often need to import data from public databases and construct their own system-specific or subject-oriented databases to support their complex analysis and knowledge discovery. To facilitate the analysis and data processing, customized and centralized databases are often created by extracting and integrating heterogeneous data retrieved from public databases. A generalized methodology for accessing, extracting, transforming and integrating the heterogeneous data is needed.

Results: This paper presents a new data integration approach named JXP4BIGI (Java XML Page for Biological Information Gathering and Integration). The approach provides a system-independent framework, which generalizes and streamlines the steps of accessing, extracting, transforming and integrating the data retrieved from heterogeneous data sources to build a customized data warehouse. It allows the data integrator of a biological database to define the desired bio-entities in XML templates (or Java XML pages), and use embedded extended SQL statements to extract structured, semi-structured and unstructured data from public databases. By running the templates in the JXP4BIGI framework and using a number of generalized wrappers, the required data from public databases can be efficiently extracted and integrated to construct the bio-entities in the XML format without having to hard-code the extraction logics for different data sources. The constructed XML bio-entities can then be imported into either a relational database system or a native XML database system to build a biological data warehouse.

Availability: JXP4BIGI has been integrated and tested in conjunction with the IKBAR system [<http://www.ikbar.org/>] in two integration efforts to collect and integrate data for about 200 human genes related to cell death from HUGO, Ensembl, and SWISS-PROT (Bairoch and Apweiler 2000), and about 700 *Drosophila* genes from FlyBase (FlyBase Consortium 2002). The integrated data has been used in comparative genomic analysis of x-ray induced cell death. Also, as

explained later, JXP4BIGI is a middleware and framework to be integrated with biological database applications, and cannot run as a stand-alone software for end users. For demonstration purposes, a demonstration version is accessible at [<http://www.ikbar.org/jxp4bigi/demo.html>].

Contact: yhuang@mail.mcg.edu; tni@cise.ufl.edu; lzhou@ufsc.c.ufl.edu; su@cise.ufl.edu

INTRODUCTION

Federated vs. data warehousing approach

A commonly recognized obstacle for data extraction and integration is that the data associated with gene, protein and other bio-entities such as regulatory pathways have been generated and maintained in different data systems (Bukhman and Skolnick 2001). The heterogeneity of these systems and their data representations have hindered the effective use of the available data (Baxevanis 2001).

Two approaches have been taken by the biological community to access and integrate data from heterogeneous databases: the federated approach and the data warehousing approach (Karp 1995; Ritter 1994; Markowitz *et al.* 1996; Letovsky 1999; Shoop *et al.* 2001; Schonbach *et al.* 2000). The federated approach builds a layer of software on top of distributed and heterogeneous databases and provides a query facility to access physically distributed data. The data warehousing approach extracts data from different databases and integrates and stores them into a centralized database for supporting complex analysis. These two approaches correspond to the virtual way and the materialized way of data integration, which is addressed in (Davidson *et al.* 1995).

Most of the current large-scale data integration projects lean towards the federated approach. For example, BioKleisli (Davidson *et al.* 1995; Chung and Wong 1999) adopts a nested relational model, extends SQL, and uses a data source driver approach to build a federated system; OPM (Markowitz, V.M. *et al.* 1996; Eckman *et al.* 2001; Chen *et al.* 1998) uses an entity

model and generic servers that retrofit and unify data sources, and provides a query language OPM-MQL to query the distributed data; IBM's DiscoveryLink (Hass *et al.* 2001) uses the relational model and the SQL language for modeling and accessing distributed data. Among the above-mentioned systems, BioKleisli and OPM can output integrated data in the XML format. The approaches taken by these systems have their own advantages and disadvantages (Shoop *et al.* 2001, Leser *et al.* 1998, Davision *et al.* 1995, Markowitz *et al.* 1995, Karp 1995, Letovsky 1999).

Although the federated approach has the advantages of providing local autonomies to existing systems, accessing up-to-date data from different data sources at run-time, and giving an integrated view for users to query distributed data, it also has the disadvantages in cost (i.e., costs in purchasing/development, installation and maintenance), performance (i.e., run-time query translation and data access from distributed data over the network), availability and reliability (i.e., possible site and network failures, making some data inaccessible).

Particularly, just retrieving a large quantity of data from those general-purpose databases is usually not very useful to a biologist. In the post-genomic era, biologists often need to selectively extract and integrate data from those general-purpose databases to form subject-oriented databases with their own data structures. Without a centralized and persistent database, it is very difficult for a biologist to perform complex follow-up analysis, computations, and simulations efficiently. Besides, a localized database will also allow the extracted data to be integrated with the biologists' own proprietary data for data analyses. The data warehousing approach can well meet the requirement.

A biological data warehouse, standing in contrast to a general-purpose database, is a subject-oriented and integrated collection of data (Schonbach *et al.* 2000). It provides a consistent and integrated environment for biologists to extract and integrate data from general-purpose databases, and performs complex follow-up analysis and knowledge discovery. Many biological systems have adopted the data warehousing approach. For example, IDG (Ritter *et al.*, 1994) integrated data retrieved from 15 source databases into one AceDB database; 3DinSight (An J, 1998) integrated PDB, SWISS-PROT and PIR sequence databases, as well as information about functional sequence motifs, enzymes, reactions, metabolites, etc.; InterPro (Apweiler *et al.* 2000) integrated data from several databases of protein domains and sequence motifs; BioMolQuest (Bukhman and Skolnick 2001) imported PDB, SWISS-PROT, Enzyme and CATH databases; PEDANT (Frishman *et al.*

2001) integrated both functional and structural information for 80 genomic sequences from external biological data sources., IXDB (Leser *et al.* 1998) integrated genomic data of the human X chromosome; MGI (Eckman *et al.* 1998) integrated genomic data into a Sybase database from external sources, such as GDB, NCBI, etc.; MetaFam (Shoop *et al.* 2001) is an integrated data warehouse which stores information about protein families and their sequences in an Oracle database. SLAD (Schonbach *et al.* 2000) is another database demonstrating a comprehensive data-warehousing integration environment.

In the data warehousing approach, most of the development efforts are on the retrieval of pertinent data from a large number of heterogeneous biological databases, followed by the extraction and transformation of the retrieved data, and eventually the integration and the loading of the integrated data into a mission-oriented, customized database. Most existing systems apply some forms of hard coding to deal with the idiosyncrasies of heterogeneous data sources during the processes of accessing, parsing and extracting data from different databases. This is very costly because different programs have to be written for different data sources and the developed program usually cannot be reused for accessing data from a different database. Furthermore, redundant program development efforts are carried out by different biological organizations. Some systems avoid hard coding by using specific scripts or formats to parse data. For example, IXDB parses data retrieved from each data source into an intermediate format IACE, then converts and stores the data into an IXDB database (Leser *et al.* 1998). BioMolQuest uses Perl scripts to guide the detailed parsing of data retrieved from each data source (Bukhman *et al.* 2001). KEGG (Ogata *et al.* 1999) provides tools for parsing and extracting data from an external database like GenBank. PEDANT develops wrappers for importing data from different external data sources (Frishman *et al.* 2001). SRS achieves a better solution by adopting objects and rules for text parsing in its own scripting language, Icarus (Zdobnov *et al.* 2002). However, almost all of the existing warehousing-type systems are system-dependent in the sense that their integration facilities are developed specifically for their systems and the extracted and integrated data are in formats suitable only for their own use. The facilities are not for general applications and data are not in a standard format for use by other systems.

We believe that a generalized solution to the warehousing-type of data integration should provide a

generalized methodology and mechanism for carrying out the steps of accessing, extracting, transforming and integrating the data retrieved from distributed data sources.

JXP4BIGI approach and its integration strategy

JXP4BIGI provides a general framework for data extraction and integration for building a biological data warehouse. The framework consists of four key components, as shown in Figure 1: XML bio-entity templates, SQL-based queries and extraction logics, generalized wrappers for accessing distributed data sources, and a JXP Processor to schedule, interpret, and execute XML templates to construct XML bio-entities.

An XML bio-entity template, also called a JXP (Java

access data of a specific format. It can be used to access data from different data sources that can provide data in the same format. The high-level specification of queries and extraction logics and the generalized wrappers allow the extraction of desired data without hard coding.

JXP4BIGI is a system-independent framework (i.e., application-independent and platform-independent) designed for use by data integrators rather than application users. It provides the software and tools for data extraction and integration and for constructing desired bio-entities. The constructed bio-entities are in a neutral data representation (i.e., XML). They can be imported into a relational database management system by using a market-available XML/DB tool, or directly stored into a native XML database system. JXP4BIGI does not dictate the underlying DBMS used to store and

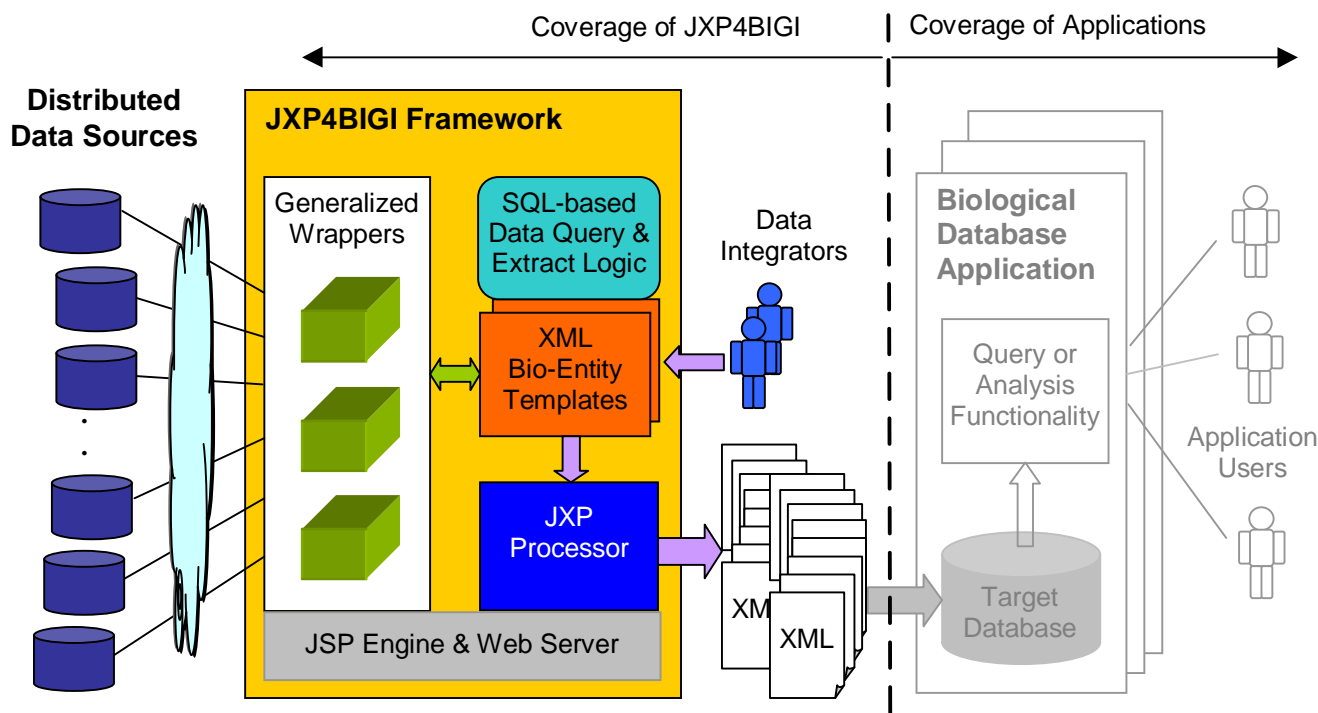


Fig. 1 The framework of JXP4BIGI

XML Page), is an XML data structure for representing a bio-entity to be constructed by a biologist. Since XML is extensible and an XML data structure is very flexible, a data integrator can easily customize the data structure of a bio-entity (e.g., a gene or protein) that he/she intends to construct out of the data extracted from different data sources. In each template, the integrator can specify SQL-based queries and extraction logics for accessing and extracting data values from heterogeneous data sources for the attributes of the customized data structure. A number of generalized wrappers are developed to “wrap” the heterogeneous data sources. Each wrapper is used to

process the constructed bio-entities. The application users can thus use the query language (e.g., SQL/XQL/XQuery) and the database management functionalities provided by the DBMS to process the bio-entities. Also, JXP4BIGI takes full advantages of the Java Server Page (Avedal and Ayers 2000), JavaBean and Web server technologies to achieve the “Write Once Run Anywhere” feature.

SYSTEM AND TECHNIQUES

How it works

JXP4BIGI uses a Java XML Page (JXP) as a template to specify an XML structure with tags to describe the structural properties of a bio-entity that a user wants to construct. A JXP is a Java Server Page (JSP) with an XML format. An example is given in Figure 2. SQL queries with range and pattern specifications and simple JavaBean-based API calls are embedded in the XML tags to specify where and how data items are to be queried and extracted from various source databases. A *source database* is an existing biological database system which contains the source data. Each source database can provide a number of *source data objects* in the form of text files, HTML pages, relational tables, or some other format/structure, and an access mechanism such as HTTP, FTP or SOAP. The left side of Figure 2 shows a sample JXP for a gene entity. Its contents are extracted from

data sources may provide data with the same format, the same generalized wrapper can be used to access and extract data from these different sources. JXP4BIGI provides a set of built-in wrappers. As long as a source data object falls into one of the recognized format types, the integrator only needs to configure some properties and parameters for the source data object (see the section Configuration of source data objects). JXP4BIGI will automatically create a proper wrapper instance that references the source database name and recognizes the format of the source data object.

A wrapper is able to parse the data retrieved from a source data object and process an SQL query against the data contents to extract the desired data values. JXP4BIGI extends SQL's query specification capability by using the dot-notation used in object-oriented query languages as well as SQL3 (Sun Microsystems 1995-2002)(Kulkarni 1994) for querying nested data structures. A nested structure can be a field containing

<pre><?xml version = '1.0'?> <GENE> <GN_ID><%=DSMgr.getTargetID0%></GN_ID> <GN_NAME> <%= DSMgr.getString("SELECT Symbol from HUGO.NomeIDs")%> </GN_NAME> <SYNONYMS> <%DSResult Literature_Aliases = DSMgr.query ("SELECT Literature_Aliases FROM HUGO.Literature_Aliases"); if(Literature_Aliases.rows>=0) for(int i=1; i<= Literature_Aliases.rows; ++i) { %> <SYNONYM><%=Literature_Aliases.getString(i,1)%></SYNONYM> <% } %> </SYNONYMS> <ORGANISM>H.Sapiens</ORGANISM> <CHROMOSOME_LOC><%=DSMgr.getString ("SELECT chrom_band FROM Ensembl.Chromosome")%> </CHROMOSOME_LOC> <GN_INFO> <SEQUENCE><%= DSMgr.getString ("SELECT seq from Ensembl.Sequence EXTRACT seq=range(2=#)")%> </SEQUENCE> <GN_STRUCTS> </GN_STRUCTS> <GN_ACCNS> <GN_ACCN> <SRC>Ensembl</SRC> <ID_REF><%=DSMgr.getString ("SELECT gene_id from Ensembl.Exons")%> </ID_REF> </GN_ACCN> </GN_ACCNS> </GN_INFO> </GENE></pre>	<pre><?xml version = '1.0'?> <GENE> <GN_ID>G10003</GN_ID> <GN_NAME>NOL3</GN_NAME> <SYNONYMS> <SYNONYM>NOL3</SYNONYM> <SYNONYM>ARC</SYNONYM> </SYNONYMS> <ORGANISM>H.Sapiens</ORGANISM> <CHROMOSOME_LOC>q23.2</CHROMOSOME_LOC> <GN_INFO> <SEQUENCE>AATACAGGCATACCACAGAGATTACAGG TTGTGTTCCAGACCACCACAATAAAGTGAGTGAGTCA CATGAATTTTTGGCTTCCAGTGCCATACAGAAGTTAT GTTACACTATACTGTAGGTTGTGTATAATAGCATTAT GTGTAAAAAATGTACATAATTAATAAACACTTTATTGC TAAAAACACTAATGATCATATAAGCCCTTCAGC..... </SEQUENCE> <GN_STRUCTS> <GN_STRUCTURE> <ES_ID>G10003E001</ES_ID> <COORDINATES>8000.8683,9181.9875, 13952.14123,17524.17708,19280.19419, 21807.21884,3839.23950,25166.25268, 8607.28663,30889.31002,33545.33730, 43587.43801,48699.48807 </COORDINATES> </GN_STRUCTURE> </GN_STRUCTS> <GN_ACCNS> <GN_ACCN> <SRC>Ensembl</SRC> <ID_REF>ENSG00000000457</ID_REF> </GN_ACCN> </GN_ACCNS> </GN_INFO> </GENE></pre>
--	---

Fig. 2 Sample JXP and generated XML page for gene entity in IKBAR system

HUGO (Human Gene Nomenclature Database) (HUGO 2002), which provides a Web-based access to human genes in tab text files, and Ensembl (Ensemble 2002; Hubbard *et al.* 2002), which provides a Web-based interface for users to query for genomic data of human, mouse, fly, etc., in a variety of formats. The right side of Figure 2 shows the constructed XML page.

Wrapping and querying structured data

One of the key ideas of JXP4BIGI is that, instead of programming for hundreds of data sources, we develop a generalized wrapper for each data format. Since different

either a nested object (Struct type in SQL3, or an Object type in Oracle 8i), or a nested table (the Array type in SQL3, or the Collection type in Oracle 8i). For example, Figure 3 shows a piece of EMBL formatted feature data of a gene entry retrieved from an Ensembl database. The feature data contains a number of sub-fields, such as source, CDS, and a number of exons. The query logic for querying exons, as shown below, will return an array of exons information:

```
SELECT EG.FT.exon FROM Ensembl.gene EG
WHERE AC='ENSG00000000457'
```

The above SQL statement selects exons in the FT field of the gene entity 'ENSG00000000457' from the Ensembl

AC	ENSG00000000457;
.....	
FT	source 1..40808
FT	/organism="Homo sapiens"
FT	CDS join(complement(35588..35752),
FT	complement(25546..25731),
FT	complement(22890..23003),
FT
FT	/db_xref="GO:GO:0005524"
FT
FT	exon complement(1..684)
FT	/exon_id="ENSE00000814455"
FT	/start_phase=0
FT	/end_phase="-1"
FT	exon complement(1182..1876)
FT	/exon_id="ENSE00000789668"
FT	/start_phase=1
FT	/end_phase=0
FT

Fig. 3 Example feature data in a nested structure

source database, where EG is an alias of Ensembl. The statement is interpreted and executed by an EMBL-format wrapper that encapsulates the source data. Different format wrappers use different algorithms to parse and manipulate the data they encapsulate when they process SQL queries. For example, the EMBL wrapper uses the format parser of BioJava (BioJava 2001). For an XML-formatted data source, the XML wrapper parses and accesses elements by using Java DOM or SAX for XML. For flat tab table format, the tab-table wrapper parses and stores the tab text in a two-dimensional data structure. If a source database allows direct access to its underlying RDBMS, a JDBC (Java Database Connection) (JDBC 1995-2002) wrapper would forward SQL query statements to the underlying system to process the source database. In this case, JXP4BIGI allows the system-specific SQL language to be specified in a JXP.

Data extraction from semi- and unstructured data

SQL queries with dot-notation work well for processing structured data. However, we are often faced with semi-structured or unstructured data that do not have well-defined schemas (Leser *et al.* 1998). Most systems that provide data with flat file formats, such as EMBL (Baker *et al.* 2000; EMBL 2002), SWISS-PROT (Bairoch and Apweiler 2000), Genbank (Benson *et al.* 2000), and DDBJ, contain semi-structured data. For example, each exon that we retrieve from the FT field of the example shown in Figure 3 actually contains 4 lines of text strings. Suppose a biologist is only interested in the data pairs that

specify exon locations, these data pairs need to be extracted from the text strings. To facilitate this, we further extend the SQL language by introducing an EXTRACT clause to specify the range and the pattern of data in a semi-structured or unstructured data block from which the desired data are to be extracted. The syntax of the extended SQL statement is as follows:

```
SELECT field1, field2,... FROM data-source-objects
EXTRACT field1=range(...)pattern(...) ...WHERE...
```

The EXTRACT clause consists of two parts: a range specification to specify the boundary of the data block of a source data object from which data is to be searched, and a pattern specification to give the specific *pattern* to be matched against the text string within the specified range from which data is to be extracted. For the example for exons shown in Figure 3, the extended SQL query is as follows:

```
SELECT EG.FT.Exon FROM Ensembl.gene EG
EXTRACT EG.FT.Exon= range(1=>1)
pattern(@{%complement([?][?])});
```

The range specification "1=>1" means to search the string only on the first line. The pattern "@{%complement([?][?])}" specifies that two data fields starting with "%complement(" and delimited by "." are to be extracted as an exon location. The result returned by the query will be a set of exon locations.

Figure 4 shows the detailed syntax of the range and pattern specifications. The symbol [?] surrounded by some delimiting characters or strings indicates the portion of the data block that is to be extracted. A special notation {[?]*match string*} in a pattern is used to express that a set of repeated and a variable number of data fields are to be extracted. The range and pattern

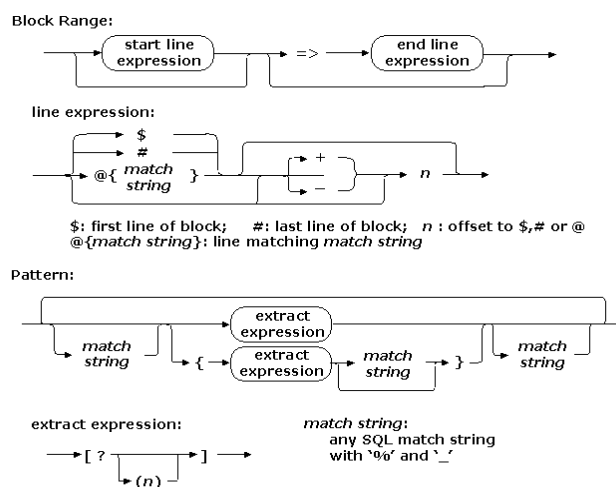


Fig.4 Syntax for block range and pattern specifications

specifications are extensively used for processing HTML-formatted data because HTML tags are for the purpose of data presentation rather than for structuring the data contained in an HTML page.

Configuration of source data objects

Making data sources configurable is the key for avoiding hard coding a wrapper for each data source. As aforementioned, the use of a generalized wrapper for each format type can significantly reduce the number of wrappers. However, such a generalized wrapper still needs to know some properties about each source data object that is to be accessed. JXP4BIGI uses a property configuration mechanism to allow the integrator to configure all the required properties and parameters for each source data object. Upon the first access of a source data object, an instance of a format wrapper is generated and initialized for the data object by reading its properties from a data source configuration file. The behavior of the wrapper is driven by the contents of the configuration file, which can be easily changed. There are four categories of properties that describe a source data object, as shown below:

a. Content and mapping information:

- content = Single/Batch is used to let the wrapper know that the source object contains either one or multiple source entries.
- map_id = a primary key or mapping ID is used to identify the connection or relationship between target entries and source entries.

b. Format: specifies the format of the source data object

- format = FreeText/TabText/TabHTML/HTML/XML/EMBLText/EMBLHTML/WebService...

c. Source and protocol: specify where the content comes from and what the access protocol is.

- protocol = HTTP/FTP/file/SOAP/JDBC...
- url = www...; or ftp...; or c:\file_dir,...

d. Method and parameters: specify the access methods for the protocol as well as the required parameters for the protocol.

- method = Get / Post (for http) or whatever other methods can be used for other protocols.
- parameter.name1 = value1

IMPLEMENTATION

Detailed architecture

JXP4BIGI provides a general framework as well as a toolkit to support the warehousing-type of data integration. Figure 5 shows the detailed architecture of the system. The JXP4BIGI framework runs in a Web-based environment with the support of Java Servlet and JSP. Major components run on the server side and within the Servlet and JSP engines. When an integrator

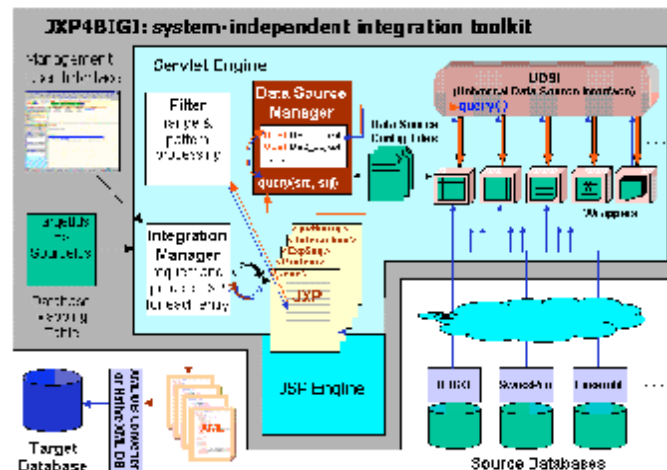


Fig. 5 Detailed architecture of JXP4BIGI

starts an integration task from a browser-based user interface, the required parameters to run the task, including a JXP page file name and a database mapping file name (the file is used for specifying the biological entry ID mappings between source databases and the target system), are forwarded to the *Integration Manager*, which initiates an integration process on the Web server side. The manager accesses the *Database Mapping Table* file, and repeatedly processes each row of the table for each of the target data entries. The *Integration Manager* requests the corresponding JXP page for each entry by passing the target and source entry IDs into the page. If this is the first request, the JXP page will be automatically translated into a Servlet by the JSP engine. Then the Servlet will run. During the execution of the JXP's Servlet, if a query in the JXP is encountered, the component *Data Source Manager* behind the JXP page will look for the proper wrapper instance based on the source database name and the source data object name specified in the query. If the wrapper instance already exists, the query will be executed. Otherwise, a new wrapper instance will be created by reading the required properties from the configuration file. A source data object from a remote or local source database will be fetched into the wrapper instance. The created wrapper instance will be stored in a wrapper instance pool maintained by the *Data Source Manager* for subsequent use. For each query, the

corresponding wrapper instance will perform the query processing and return a data set, which is in turn returned to the JXP's Servlet. The Servlet generates XML tags for the data in the returned data set, produces an XML page, and sends it to the *Integration Manager*, which stores it in a persistent store. The *Integration Manager* also reports the integration status and the result to the user through the user interface. The user can then review the XML page. A target database system can import all the constructed XML pages into its database by using an XML/DB conversion facility or store the XML pages directly into a native XML database.

JXP4BIGI has been used and tested in conjunction with the IKBAR system in two integration efforts to collect and integrate data for about 200 human genes related to cell death from HUGO, Ensembl, and SWISS-PROT (Bairoch and Apweiler 2000), and about 700 *Drosophila* genes from Flybase (FlyBase Consortium 2002). The integrated data has been used in comparative genomic analysis of x-ray induced cell death.

SUMMARY

The key features of the JXP4BIGI approach can be summarized as follows:

1) JXP4BIGI is a system-independent framework for data extraction and integration for building a biological data warehouse. It provides a number of generalized wrappers and an extended SQL with range and pattern specification capabilities for extracting data from heterogeneous data sources. Information required for identifying and accessing heterogeneous data sources can easily be specified in configuration files without having to hard-code the information in wrapper programs. The generalized wrappers are developed for different data format types instead of different source database systems.

2) The EXTRACT clause is an extension to the SQL language. It allows semi-structured and unstructured data that has been retrieved from a source database to be further processed based on range and pattern specifications. Thus, unwanted data can be effectively filtered out during the construction of customized bio-entities. The extended SQL provides users with a unified querying facility.

3) The JXP4BIGI approach allows data retrieval, extraction, transformation and integration to be carried out in a streamlined process to create bio-entities in the structures desired by biologists.

4) Since the templates used to define target bio-entities are in the XML format, they allow target entities with very complex and flexible structures to be defined. The resulting XML documents that define bio-entities can

be readily transported and shared with other biologists over the Internet. They can also be imported into a local database for further local processing and analysis.

5) By adopting Java, JSP, XML, and Web server technologies, JXP4BIGI takes full advantage of open standards and technologies developed by the IT industry. It uses the commercial JSP engine to interpret JXP template pages and to generate XML pages, thus minimizing the cost and effort of software development. By using these open standards and technologies, JXP4BIGI offers a platform-independent implementation, achieving the advantage of "write once, run anywhere".

ACKNOWLEDGEMENTS

This work is supported in part by a university research opportunity grant to Lei Zhou and Stanley Su. We thank Rong Yuan and Yuan Shi for helpful discussions and comments in the course of this work.

REFERENCES

- An J, Nakama T, Kubota Y, and Sarai A. (1998) 3DinSight: an integrated relational database and search tool for the structure, function and properties of biomolecules. *Bioinformatics*, **14**, 188-95.
- Apweiler, R., Attwood, T. K., Bairoch, A., Bateman, A., Birney, E., Biswas, M., Bucher, P., Cerutti, L., Corpet, F., Croning, M. D., Durbin, R., Falquet, L., Fleischmann, W., Gouzy, J., Hermjakob, H., Hulo, N., Jonassen, I., Kahn, D., Kanapin, A., Karavidopoulou, Y., Lopez, R., Marx, B., Mulder, N. J., Oinn, T. M., Pagni, M., Servant, F., Sigrist, C. J. and Zdobnov, E. M. (2000) InterPro--an integrated documentation resource for protein families, domains and functional sites. *Bioinformatics*, **16**(12), 1145-1150.
- Avedal, K. and Ayers, D., Briggs, T., Burnham, C., Halberstadt, A., Haynes, R., Henderson, P., Holden, M., Li, S., Malsk, D., Myers, T., Nakhimovsky, A., Osmont, S., Palmer, G., Timney, J., Tyagi, S., Damme, G.V., Wilcox, M., Wilkinson, S., Zeiger, S. and Zukowski, J. (2000) *Professional JSP*. Wrox Press., Birmingham, UK.
- Bairoch, A. and Apweiler, R. (2000) The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.*, **28**(1), 45-48.
- Baker, W., van den Broek, A., Camon, E., Hingamp, P., Sterk, P., Stoesser, G. and Tuli, M. A. (2000) The EMBL nucleotide sequence database. *Nucleic Acids Res.*, **28**(1), 19-23.
- Baxevanis, A. D. (2001) Information retrieval from biological databases. *Methods Biochem. Anal.*, **43**, 155-185.
- Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., Rapp, B. A. and Wheeler, D. L. (2000) GenBank. *Nucleic Acids Res.*, **28**(1), 15-18.
- BioJava (2001) BioJava.

-
- Bukhman, Y. V. and Skolnick, J. (2001) BioMolQuest: integrated database-based retrieval of protein structural and functional information. *Bioinformatics*, **17**(5), 468-478.
- Chen, I. M., Kosky, A. S., Markowitz, V. M., Szeto, E. and Topaloglou, T. (1998) Advanced query mechanisms for biological databases. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **6**, 43-51.
- Chung, S. Y. and Wong, L. (1999) Kleisli: a new tool for data integration in biology. *Trends Biotechnol.*, **17**(9), 351-355.
- Davidson, S. B., Overton, C. and Buneman, P. (1995) Challenges in integrating biological data sources. *J. Comput. Biol.*, **2**(4), 557-572.
- Eckman, B. A., Aaronson, J. S., Borkowski, J. A., Bailey, W. J., Elliston, K. O., Williamson, A. R. and Blevins, R. A. (1998) The Merck Gene Index browser: an extensible data integration system for gene finding, gene characterization and EST data mining. *Bioinformatics*, **14**(1), 2-13.
- Eckman, B. A., Kosky, A. S. and Laroco, L. A., Jr. (2001) Extending traditional query-based integration approaches for functional characterization of post-genomic data. *Bioinformatics*, **17**(7), 587-601.
- EMBL (2002) EMBL Features and Qualifiers. <http://www3.ebi.ac.uk/Services/WebFeat>
- Ensemble (2002) http://www.ensemble.org/Homo_sapiens/exportview
- FlyBase Consortium (2002) The FlyBase database of the Drosophila genome projects and community literature. *Nucleic Acids Res.*, **30**(1), 106-108.
- Frishman, D., Albermann, K., Hani, J., Heumann, K., Metanowski, A., Zollner, A. and Mewes, H. W. (2001) Functional and structural genomics using PEDANT. *Bioinformatics*, **17**(1), 44-57.
- Hass, L., Schwartz, P., Kodali, P., Kotlar, E., Rice, J. and Swope, W. (2001) DiscoveryLink: a system for integrating life sciences data. *IBM Syst. J.*, **40**, 489-511.
- Hubbard, T., Barker, D., Birney, E., Cameron, G., Chen, Y., Clark, L., Cox, T., Cuff, J., Curwen, V., Down, T., Durbin, R., Eyraas, E., Gilbert, J., Hammond, M., Huminiecki, L., Kasprzyk, A., Lehvaslaiho, H., Lijnzaad, P., Melsopp, C., Mongin, E., Pettett, R., Pocock, M., Potter, S., Rust, A., Schmidt, E., Searle, S., Slater, G., Smith, J., Spooner, W., Stabenau, A., Stalker, J., Stupka, E., Ureta-Vidal, A., Vastrik, I. and Clamp, M. (2002) The Ensembl genome database project. *Nucleic Acids Res.*, **30**(1), 38-41.
- HUGO (2002) The Human Genome Organisation. <http://www.gene.ucl.ac.uk/hugo/>
- JDBC (1995-2002) JDBC Data Access API. <http://java.sun.com/products/jdbc/overview.html>
- Karp, P. D. (1995) A strategy for database interoperation. *J. Comput. Biol.*, **2**(4), 573-586.
- Kulkarni, K. (1994) Object-oriented extensions in SQL3: A status report. *ACM SIGMOD Conf. on Manage. of Data*, Minneapolis, MN, USA.
- Leser, U., Lehrach, H. and Roest Crollius, H. (1998) Issues in developing integrated genomic databases and application to the human X chromosome. *Bioinformatics*, **14**(7), 583-590.
- Letovsky, S. (1999) Introduction. In Letovsky, S., (eds). *Bioinformatics Databases and Systems*. Kluwer Academic Publishers, Boston, pp. 1-7.
- Markowitz, V. M., Chen, I. A. and Kosky, A. S. (1996) Exploring Heterogeneous Molecular Biology Databases in the Context of the Object-Protocol Model. http://gizmo.lbl.gov/DM_TOOLS/OPM/OPM_QS/PAPER/OPM_QS_Paper.html
- Markowitz, V. M. and Ritter, O. (1995) Characterizing heterogeneous molecular biology database systems. *J. Comput. Biol.*, **2**(4), 547-556.
- Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H. and Minoru (1999). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*. **27**(1), 29-34.
- Ritter, O., Kocab, P., Senger, M., Wolf, D. and Suhai, S. (1994) Prototype implementation of the integrated genomic database. *Comput. Biomed. Res.*, **27**(2), 97-115.
- Schonbach, C., Kowalski-Saunders, P. and Brusica V. (2000) Data warehousing in molecular biology. *Briefings in Bioinformatics*. **1** (2), 190-198.
- Shoop, E., Silverstein, K. A., Johnson, J. E. and Retzel, E. F. (2001) MetaFam: a unified classification of protein families. II. Schema and query capabilities. *Bioinformatics*, **17**(3), 262-271.
- Sun Microsystems (1995-2002) Lesson: new features in the JDBC 2.0. <http://java.sun.com/docs/books/tutorial/jdbc/jdbc2dot0/index.html>
- Zdobnov, E. M., Lopez, R., Apweiler, R. and Etzold, T. (2002) The EBI SRS server--recent developments. *Bioinformatics*, **18**(2), 368-373.
-