BIOINFORMATICS

ISMB 2005
Pages 1–1

YeastHub: a semantic web use case for integrating data in the life sciences domain

Kei-Hoi Cheung[1,2,], Kevin Y. Yip[3], Andrew Smith[3], Remko deKnikker[1], Andy Masiar[1], Mark Gerstein[3,4]*

[1]Center for Medical Informatics, Anesthesiology, [2]Genetics, [3]Computer Science, [4]Molecular Biophyiscs and Biochemistry, Yale University, USA

ABSTRACT

Motivation: As the semantic web technology is maturing and the need for life sciences data integration over the web is growing, it is important to explore how data integration needs can be addressed by the semantic web. The main problem that we face in data integration is a lack of widely-accepted standards for expressing the syntax and semantics of the data. We address this problem by exploring the use of semantic web technologies — including Resource Description Framework (RDF), RDF Site Summary (RSS), relational-database-to-RDF mapping (D2RQ), and native RDF data repository — to represent, store, and query both metadata and data across life sciences datasets.

Results: As many biological datasets are presently available in tabular format, we introduce an RDF structure into which they can be converted. Also, we develop a prototype web-based application called *YeastHub* that demonstrates how a life sciences data warehouse can be built using a native RDF data store (Sesame). This data warehouse allows integration of different types of yeast genome data provided by different resources in different formats including the tabular and RDF formats. Once the data are loaded into the data warehouse, RDF-based queries can be formulated to retrieve and query the data in an integrated fashion.

Availability: The YeastHub web site is accessible via the following URL: http://yeasthub.gersteinlab.org

Contact: kei.cheung@yale.edu

1 INTRODUCTION

The web has become instrumental to many facets of research in the life sciences domain. Nowadays, researchers can easily have Internet access to a large quantity and variety of biological data using their web browsers running on local desktop computers. As the number of these web resources continues to increase, it is important to address the problem of interoperability. Currently, it is a challenging problem for the following reasons.

1. It is difficult to automatically identify web sites that contain relevant and interoperable data, as there is a lack of widely-accepted standards for describing these web sites as well as their contents. Although approaches like the HTML *meta* tag (http://www.htmlhelp.com/reference/html40/head/meta.html) can be used to annotate a web page through the use of keywords, they are problematic in terms of sensitivity and specificity. In addition, these approaches are neither supported nor used widely by existing web search engines. Most web search engines rely on using their own algorithms to index individual web sites based on their contents.
2. Different resources provide their data in heterogeneous formats. For example, while some data are represented in HTML format that is interpretable by the web browser, other data formats including the text format (e.g., tab-delimited files) and binary format (e.g., images) are used. Such heterogeneity in data formats makes interoperability difficult if not impossible.
3. Data interoperability involves both syntactic and semantic translation. Both types of translation are hindered by the lack of standard data models, formats, and vocabulary/ontology.

The semantic web research community addresses these problems by seeking methods to facilitate machine-based identification and semantic interoperability of web resources. Crucial to the semantic web approach is the design and development of ontologies (semantic part) that are represented in computer-readable formats (syntactic part). The eXtensible Markup Language (XML) has become a standard syntax for expressing data that are exchanged between applications. In the past several years, a large collection of XML-based formats has emerged for representing different types of biological data. Examples include mzXML (Pedrioli et al. 2004) for standardizing the representation of mass spectrometry (MS) data generated by different MS instruments, BioML (Fenyo 1999) for representing biopolymer data, MAGE-ML (Spellman et al. 2002) for representing microarray gene expression data, SBML (Hucka et al. 2003) for representation and exchange of biochemical network models, and ProML (Hanisch et al.

* To whom correspondence should be addressed.

2002) for specifying protein sequences, structures and families. In addition, since XML is widely used there are a large number and variety of open source software tools for processing it.

While these XML formats facilitate data exchange between applications, they do not adequately address semantics and lack expressivity for knowledge representation and inference (Decker et al. 2000). In addition, there is a proliferation of semantically-overlapping XML formats in the life sciences domain, making syntactic and semantic data translation more complex and difficult. For example, AGAVE (http://www.agavexml.org/) and BSML (http://www.bsml.org/) are different XML formats for describing sequence annotation. SBML, PSI-MI (Hermjakob et al. 2004), BIND XML (Alfarano et al. 2005), and BioPax (http://www.biopax.org/) are examples of pathway/network data formats. Efforts have been underway to unify some of these XML formats. For example, MAML and GEML, which were two separate microarray gene expression data formats, were consolidated into MAGE-ML.

The Resource Description Framework (RDF) is a standardized XML format designed to describe web resources. The RDF structure is generic in the sense that it is based on the directed acyclic graph (DAG) model. RDF is a model for defining statements about resources and relationships among them. Each statement is a triplet consisting of a subject, a property, and a property value (or object). For example, <"Protein" "Name" "P53"> is a triple statement expressing that the subject "Protein" has "P53" as the value of its "Name" property. RDF also provides a means of defining classes of resources and properties. These classes are used to build statements that assert facts about resources. Each resource possesses one or more properties. While the grammar for XML documents is defined using DTD or XSchema, RDF uses its own syntax (RDF Schema or RDFS) for writing a schema for a resource. RDFS is expressive and it includes subclass/superclass relationships as well as constraints on the statements that can be made in a document conforming to the schema. Unlike the order of elements in XML, the order of RDF properties does not matter, thereby giving more flexibility to web programmers in developing their applications. While RDF can be serialized to a standard XML format, other representations such as Notation3 also exist.

The generic structure of RDF makes data interoperability and evolution easier to handle as different types of data can be represented using the common graph model. RDF extensions such as the Web Ontology Language (OWL) support more sophisticated knowledge representation and inference. Such languages allow data semantics to be defined declaratively (not procedurally) and can be used as a common model for expressing different types of biological data that are currently defined using different XML syntaxes. There are already some biological data that are expressed in RDF format. Examples include Gene Ontology (Ashburner et al. 2000), NCI thesaurus (Goldbeck et al. 2003), and UniProt (Apweiler et al. 2004).

As RDF is gaining more attention in the bioinformatics community and more RDF-related tools and technologies are becoming available, it is important to find new use cases of RDF in the life sciences domain (http://www.w3.org/2004/07/swls-ws.html). To this end, our paper demonstrates how to use RDF metadata/data standards (e.g., RDF Site Summary or RSS) and RDF-based technologies (e.g., native RDF database) to facilitate integration of diverse types of genome data provided by multiple web resources in heterogeneous formats. This builds upon our previous work on using XML to interoperate heterogeneous genome data (Cheung et al. 2004).

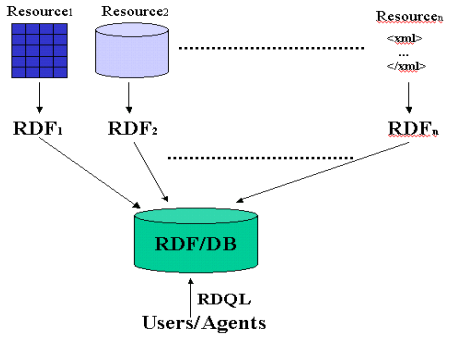

Fig. 1. System overview.

2 RDF DATA WAREHOUSE

Fig. 1 gives a system overview of our semantic web approach to data integration. It entails the following steps.

1. Describing and downloading the contents (as tab-delimited or RDF files) from individual genome web sites.
2. Converting the downloaded data into our RDF format if these data are in tab-delimited format. If the data files are in RDF format (even though they are different from our RDF format), no conversion is required.
3. Loading the RDF-formatted data files into an RDF-native database for data storage, management, and retrieval. Once the data are stored in a repository, (web-enabled) applications can be written to allow users to access, query, and analyze the data.

For data that are already stored in relational databases, we explore a relational-database-to-RDF mapping method, D2RQ (http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rq/), which allows existing (or legacy) relational databases to publish data in RDF format via a high-level mapping specification language.

2.1 RDF data stores

While relational database management systems (RDBMS) are the predominant platform for storing, managing, and querying biological data, they do not directly fit the RDF structure that is based on the DAG model. Mapping methods or new database engines are needed to handle RDF datasets efficiently. Given the growing use of RDF, specialized data storage methods (called "triple stores") have been developed to efficiently store, manage, and query RDF-formatted data. Representative approaches include: Sesame (http://www.openrdf.org), Kowari (http://www.kowari.org), Joseki (http://www.joseki.org), and Triplestore (http://triplestore.aktors.org).

Some data store approaches (e.g., Sesame) use or provide the option to use a relational database (e.g., Oracle, MySQL, and Postgres) as the underlying persistent store. Others (e.g., Kowari) allow a repository to be created directly on top of the RDF files without the need of using a relational database. Many of these RDF database systems come with their own implementation of RDF query languages (e.g., SeRQL is implemented by Sesame and iTQL by Kowari).

A scalability report on existing RDF data stores has been published (http://simile.mit.edu/reports/stores/). In the report, Sesame and Kowari are rated high in terms of their performance, ease of use, and deployment. Based on this report, we have made the decision to use Sesame to implement the data warehouse. In addition, Sesame is the only system that allows main memory, relational database, and file approaches to be used to construct a repository. This lets us compare these underlying storage approaches.

2.2 Metadata and data

In our system, each resource has two RDF files created and associated with it, metadata and data. Fig. 2 shows the first step of entering information needed to generate the metadata. Based on the information entered, our system will generate metadata in RDF format. The RDF format that we use is based on the RDF Site Summary (RSS; http://web.resource.org/rss/1.0/), which is a standard format originally intended for sharing news headlines and contents between web sites. In RSS terms, each resource is known as a **channel**. The basic idea of RSS is that each news web site will publish (or syndicate) its headline and description of its contents as an **RSS feed**; applications such as **aggregators** can spider these RSS-enabled sites and assemble their feeds. We use a similar idea to create and store the genome-oriented RSS feeds centrally. Our data warehouse system can be considered as an aggregator that integrates the data that are described in the RSS feeds.

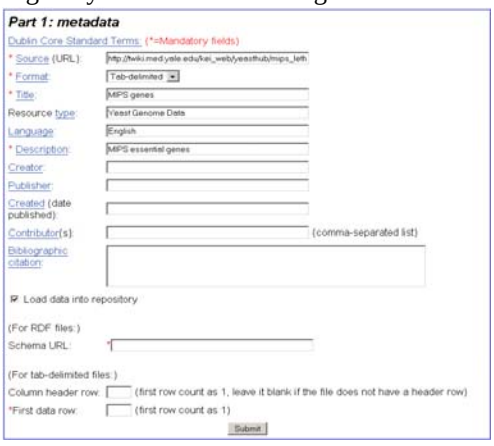

Fig. 2. Metadata generation step.

The RSS format we use incorporates different sets (or modules) of vocabularies including the Dublin Core Metadata (DCM) vocabulary (http://dublincore.org/documents/dcmi-terms/). We use the following DCM terms/properties.

1. **Source URL** gives the web address or URL through which the original data resource (or channel) can be accessed. In our case a resource or channel is a particular data file.
2. **Format** indicates the format of the original data file: tab-delimited and RDF.
3. **Title** is a descriptive name given to a resource.
4. **Type of resource** is a list of types that can be used to categorize the nature of the content of the resource.
5. **Language** indicates the language in which the resource contents are published.
6. **Description** gives an account of the resource content.
7. **Identifier** is used to identify a resource uniquely. Our system generates this identifier automatically and assigns it to the *identifier* property.
8. **Creator** indicates the entity (e.g., a person, organization, or service) that is responsible for making the original resource available.
9. **Publisher** indicates the entity (e.g., a person, organization, or service) that makes the resource that is derived from the original resource available.
10. **Created** indicates the date on which the original resource is created.
11. **Contributor** identifies the individual(s) who makes contribution to the content of the resource.
12. **Bibliographic citation** gives a bibliographic reference to the resource.

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rss="http://purl.org/rss/1.0/">
    <rss:channel rdf:about="http://twiki.med.yale.edu/kei_web/yeasthub/mips_lethal_genes2.txt">
            <rss:link>http://twiki.med.yale.edu/kei_web/yeasthub/mips_lethal_genes2.txt</rss:link>
            <dc:identifier>58639</dc:identifier>
            <dc:format>tabDelimited</dc:format>
            <rss:title>MIPS genes</rss:title>
            <dc:type>Yeast genome data</dc:type>
            <dc:language>English</dc:language>
            <rss:description>MIPS essential genes</rss:description>
            <rss:items>
                    <rdf:Seq>
                            <rdf:li rdf:resource="http://mcdb750.med.yale.edu/yeasthub/data/datanull.rdf" />
                    </rdf:Seq>
            </rss:items>
    </rss:channel>
    <rss:item rdf:about="http://mcdb750.med.yale.edu/yeasthub/data/datanull.rdf">
            <rss:link>http://mcdb750.med.yale.edu/yeasthub/data/datanull.rdf</rss:link>
            <rss:title>MIPS genes</rss:title>
    </rss:item>
</rdf:RDF>
```

Fig. 3. Metadata encoded in RSS 1.0 format.

While **title**, **description**, **identifier** (generated by the system) and **source URL** are mandatory, the other properties are optional. By using these standard properties, we hope to broaden the utility and sharing of metadata. Fig. 3 gives an example of the metadata represented using these properties in RSS format.

If the source data file is in RDF format, the user just needs to provide the URL of the corresponding schema file. If the source data file is in tabular format, the user needs to indicate whether the data file contains column headers and if so, at which line they occur. Also, the user needs to indicate the line number of the first data row. In addition to data conversion, the user is offered the option to load the converted dataset into the RDF repository for storage and later query retrieval; queries can be done not only for the just stored dataset, but also integrated queries over all resources stored in the repository can be done.

During the second step of data registration — data generation (as shown in Fig. 4), the user needs to provide information on how the RDF data format should be generated based on the tabular structure (as shown on top of Fig .4). This is divided into two parts.

1. The first part requires the user to indicate the type of genome objects and the organism involved. In addition, the user needs to enter the default namespace for the properties to which the file columns (headers) are mapped (see below). Finally, the user indicates which column (if any) is the ID column by entering the corresponding URL, which includes the string pattern "[ID]" that will be replaced by the actual ID value.
2. In the second part, the property name is entered for each file column selected by the user. If the source file contains column headers, the header labels will be used as the default property names (which can later be edited by the user). It is possible that the properties may have been defined in schemas identified by different namespaces. Therefore, the interface provides the user with the option to enter a namespace for each property. In addition, the interface lets the user indicate whether a single column entry contains multiple values (e.g., gene synonyms separated by "|"). If so, the user has to indicate the delimiter (e.g., comma or space) that is used to separate the values. In this case, the corresponding RDF output will have multiple property-value pairs. This may simplify data querying later. Finally, the interface allows the user to replace a substring pattern with another substring pattern when converting column values to property values. For example, a GO ID in one resource may contain a colon (e.g., GO:12345), while in another resource it has no colon (e.g., GO12345). Such a substring replacement function helps reduce data variability between resources, thereby easing data integration.

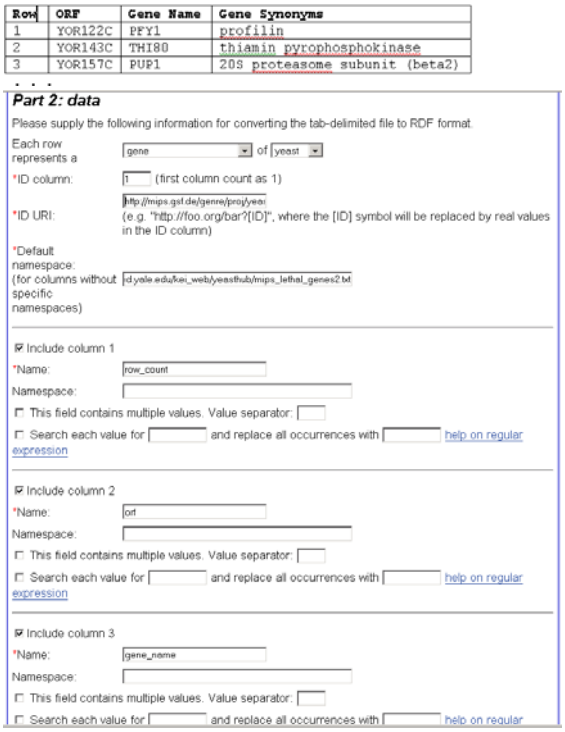

Fig. 4. Tabular-to-RDF data conversion.

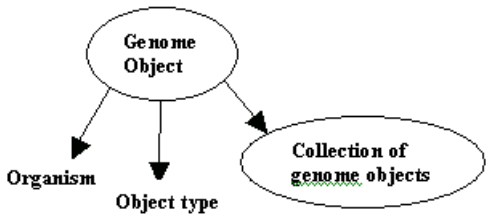

Fig. 5. Class diagram of the YeastHub RDF data model.

Currently, our RDF conversion applies only to data that are represented in tab-delimited format. In addition to converting tab-delimited files into RDF format, our system generates the corresponding RDF schema. Fig. 5 depicts the RDF schema generally. In the figure, there is a class named **genome object** that is associated with a collection of individual genome objects (a collection is a special type of RDF container). Also, **genome object** has the properties,*object type* and *organism* which describe the type of the genome objects involved (e.g., genes, markers, or proteins) and the organism of interest (e.g., yeast, human, or mouse). Each genome object in the collection can be described by a set of properties that can be user-defined or derived from existing standard vocabularies. Different collections of genome objects (obtained from different sources) may involve different sets of properties.

Fig. 6 illustrates how a collection of yeast genes is expressed in our RDF/XML format. In this example, the description of each yeast gene includes the standard open reading frame (ORF) name, common gene name, and

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:yh="http://mcdb750.med.yale.edu/yeasthub/schema/yeasthub_schema.rdf"
xmlns:ns0="http://mcdb750.med.yale.edu/yeasthub/schema/schema58639.rdf">
 <rdf:Description rdf:about="http://twiki.med.yale.edu/kei_web/yeasthub/mips_lethal_genes2.txt">
   <yh:object_type>gene</yh:object_type>
   <yh:organism>yeast</yh:organism>
   <yh:genome_objects rdf:parseType="Collection">
     <rdf:Description
rdf:about="http://mips.gsf.de/genre/proj/yeast/searchEntryAction.do?text=YAL001Cc&amp;db=CYGD">
       <ns0:row_count>1</ns0:row_count>
       <ns0:orf>YAL001C</ns0:orf>
       <ns0:gene_name>TFC3</ns0:gene_name>
       <ns0:gene_synonyms>TFIIIC (transcription initiation factor) subunit, 138 kD</ns0:gene_synonyms>
     </rdf:Description>
     <rdf:Description
rdf:about="http://mips.gsf.de/genre/proj/yeast/searchEntryAction.do?text=YAL003Wc&amp;db=CYGD">
       <ns0:row_count>2</ns0:row_count>
       <ns0:orf>YAL003W</ns0:orf>
       <ns0:gene_name>EFB1</ns0:gene_name>
       <ns0:gene_synonyms>translation elongation factor eEF1beta</ns0:gene_synonyms>
     </rdf:Description>
     ...
>/rdf>
```

Fig. 6. An example collection of yeast essential genes represented in RDF/XML format.

synonyms. Each gene is identified by a URL that takes the ORF name as a parameter and returns the detailed description of the gene from MIPS.

The link connecting the generated RDF metadata file, data file, and the schema file is via a common system-generated identifier that is stored in the property *identifier* in the metadata file. We create a RDF repository for each file type.

3 BIOLOGICAL USE CASE: YEASTHUB

To demonstrate how to use semantic web techniques to integrate diverse types of genome data in heterogeneous formats, we have developed a prototype application called "YeastHub". In this application, a data warehouse has been constructed using Sesame to store and query a variety of yeast genome data obtained from multiple sources. For performance reasons, we create the RDF repository using main memory. The application allows the user to register a dataset and convert it into RDF format if it is in tabular format. Once the datasets are loaded into the repository, they can be queried in the following ways.

1. **Ad hoc queries.** This allows the user to compose RDF-based query statements and issue them directly to the data repository. Currently, it allows the user to use the following query languages: RQL, SeRQL, and RDQL. This requires the user to be familiar with at least one of these query syntaxes as well as the structure of the RDF datasets to be queried. SQL users should find it easy to learn RDF query languages.
2. **Form-based queries**. While ad hoc RDF queries are flexible and powerful, users who do not know RDF query languages would prefer to use an alternative method to pose queries. Even users who are familiar with RDF query languages might find these languages arcane to use. To this end, the application allows users to query the repository through web query forms (although they are not as flexible as the ad hoc query approach). To create these query forms, YeastHub provides a query template generator. Fig. 7 shows the

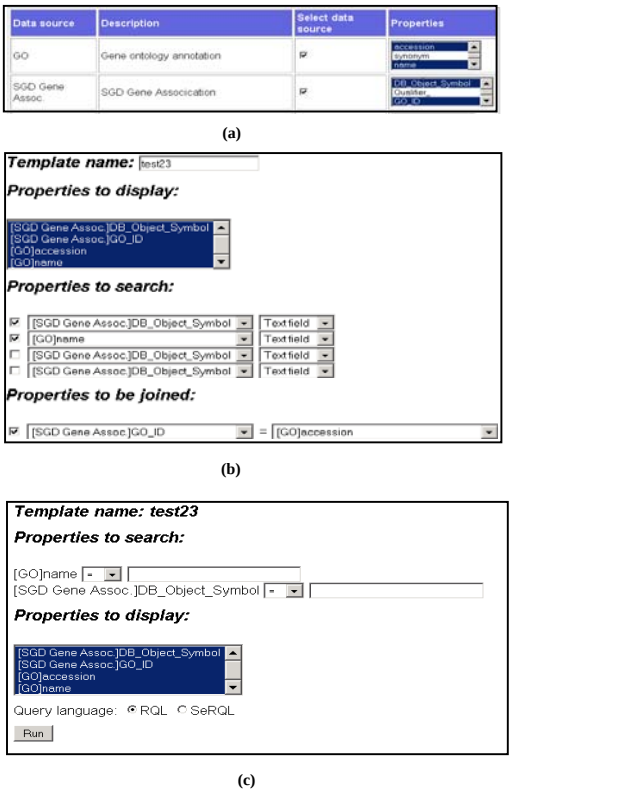

Fig. 7. (a) Selection of data sources and properties for creating a query template. **(b)** Query template generation. **(c)** Generated query form template.

web pages that allow the user to perform the steps involved in generating and saving the query form. First, as shown in Fig. 7 (a), the user selects the datasets and the properties of interest. After the selection, the user proceeds to specify how to generate the query form template, as shown in Fig. 7 (b). This page requires the user to indicate which properties are to be used for the query output (select clause), search Boolean criteria (where clause), and join criteria. In addition, the user is given the option to create a textfield, pulldown menu, or select list (in which multiple items can be selected) for each search property. Once the entry is complete, the user can go ahead to generate the query form by saving it with a name (all this information is stored as metadata in a MySQL database). The user can then use the generated query form, as shown in Fig. 7(c), to perform Boolean queries on the selected datasets. Notice that the user who generates the query is not necessarily the same person who uses the form to query the repository. Some users may just use the query form(s) generated by someone else to perform data querying. These users may not have the need to create query forms themselves.

Presently, both types of queries return results in HTML format for display to the human user. Other formats (e.g., RDF format) can be provided.

3.1 Example Queries

Our example queries involve integrating datasets obtained from different web-accessible databases. Table 1 lists these databases. In addition to showing the data distribution formats, it categories the databases into the following types.

1. **Global databases** represent very large repositories typically consisting of gigabytes or terabytes of data. These databases are widely accessed by researchers from different countries via the Internet. The example here is the yeast portion of UniProt in RDF format.
2. **Boutique databases** are large databases with typical sizes ranging from several megabytes to hundreds of megabytes (or even several gigabytes). Examples include SGD, YGDP, MIPS, BIND, GO, and TRIPLES. While SGD and MIPS datasets are typically available in tabular format, GO and BIND are available in XML format. TRIPLES is a relational database.
3. **Local databases** are relatively small databases that are typically developed and used by individual laboratories. These databases may range from several kilobytes to several (or tens of) megabytes in size. Examples include a protein-protein interaction dataset extracted from BIND and a protein kinase chip dataset. While global and boutique databases are mostly Internet-accessible, some local databases may be network-inaccessible and may involve proprietary data formats.

Table 1. Types of databases and data distribution formats.

	Tabular	XML	RDF	Rel. DB
Global Databases (GB/TB)		BIND	UniProt	
Boutique Databases (MB/GB)	SGD, YGDP, MIPS	GO		TRIPLE
Local Databases (KB/MB)	Protein Chips, Protein-Protein Interactions			

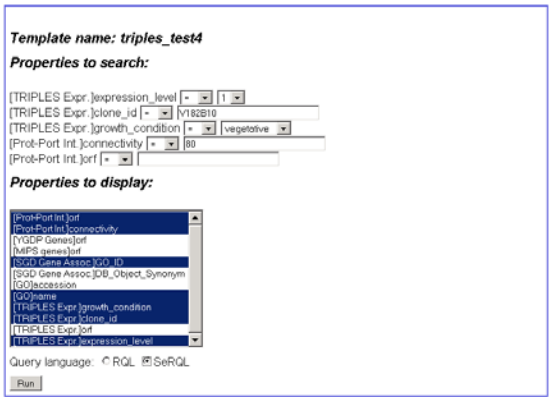

Fig. 8. Example integrated query form.

```
SELECT DISTINCT ns0orf,ns0connectivity,ns4accession,ns4name,ns5growth_condition,
ns5clone_id, ns5expression_level
FROM
{source58640} ns1:orf {ns1orf},
{source58639} ns2:orf {ns2orf},
{source58638} ns3:DB_Object_Synonym {ns3DB_Object_Synonym},
{source58638} ns3:GO_ID {ns3GO_ID},
{source58636} ns4:name {ns4name},
{source58636} ns4:accession {ns4accession},
{source55396} ns5:orf {ns5orf},
{source55396} ns5:growth_condition {ns5growth_condition},
{source55396} ns5:expression_level {ns5expression_level},
{source55396} ns5:clone_id {ns5clone_id},
{source58642} ns0:connectivity {ns0connectivity},
{source58642} ns0:orf {ns0orf}
WHERE
ns0connectivity="80"
AND ns5expression_level="1"^^<http://www.w3.org/2001/XMLSchema#longInteger>
AND ns5clone_id="V182B10"^^<http://www.w3.org/2001/XMLSchema#string>
AND ns5growth_condition="vegetative"^^<http://www.w3.org/2001/XMLSchema#string>
AND ns0orf=ns1orf
AND ns1orf=ns2orf
AND ns2orf=ns3DB_Object_Synonym
AND ns3DB_Object_Synonym=ns5orf
AND ns3GO_ID=ns4accession
USING NAMESPACE
ns2=<http://mcdb750.med.yale.edu/yeasthub/schema/schema58639.rdf> ,
ns3=<http://mcdb750.med.yale.edu/yeasthub/schema/schema58638.rdf> ,
ns1=<http://mcdb750.med.yale.edu/yeasthub/schema/schema58640.rdf> ,
ns0=<http://mcdb750.med.yale.edu/yeasthub/schema/schema58642.rdf> ,
ns5=<http://mcdb750.med.yale.edu/yeasthub/schema/schema_triples.rdf#> ,
ns4=<http://139.91.183.30:9090/RDF/VRP/Examples/schema_go.rdf>
```

ns0orf	ns0connectivity	ns4accession	ns4name	ns5growth_condition	ns5clone_id	ns5expression_level
YBL092W	80	GO:0005842	cytosolic large ribosomal subunit (sensu Eukaryota)	vegetative	V182B10	1
YBL092W	80	GO:0003735	structural constituent of ribosome	vegetative	V182B10	1
YBL092W	80	GO:0006412	protein biosynthesis	vegetative	V182B10	1

Fig. 9. Syntax and output of example query 1.

Example Query 1: Figure 8 shows a query form that allows the user to simultaneously query the following yeast resources: a) essential gene list obtained from MIPS, b) essential gene list obtained from YGDP, c) protein-protein interaction data (Yu et al. 2004), d) gene and GO ID association obtained from SGD, e) GO annotation and, f) gene expression data obtained from TRIPLES (Kumar et al. 2002). Datasets (a)- (d) are distributed in tab-delimited format. They were converted into our RDF format. The GO dataset is in an RDF-like XML format (we made some slight modification to it to make it RDF-compliant). TRIPLES is an Oracle database. We used D2RQ to dynamically map a subset of the gene expression data stored in TRIPLES to RDF format.

The example query demonstrates how an integrated query can be used to correlate between gene essentiality and connectivity derived from the interaction data. The hypothesis is that the higher its connectivity, the more likely that the gene is essential. This hypothesis has been investigated in other work (Guelzim et al. 2002; Wuchty 2004). In the query form shown in Fig. 8, the user has entered the following Boolean condition: *connectivity = 80*, *expression_level = 1*, *growth_condition = vegetative*, and *clone_id = V182B10*. Such Boolean query joins across six resources based on common gene names and GO IDs. Fig. 9 shows the corresponding SeRQL query syntax and output. The query output indicates that the essential gene (YBL092W) has a connectivity equal to 80. This gene is found in both the MIPS and YGDP essential gene lists. This gives a higher confidence of gene essentiality as the two resources might have used different methods and sources to identify their essential genes. The query output displays GO annotation (molecular function, biological process, and cellular component) and TRIPLES gene expression.

Example Query 2: This query demonstrates how to integrate the UniProt dataset with the yeast protein kinase

chip dataset that captures the number of substrates that each kinase phosphorylates with an expression level > 1. Fig. 10 shows the RQL query syntax and the output that gives the number of substrates phosphorylated by kinase "YBL105C" (level >1) as well as the functional annotation of the kinase. This protein is listed as essential in both MIPS and YGDP. In addition to connectivity, we might hypothesize that the more the number of substrates a kinase phosphorylates at a high level, the more likely that the kinase is essential.

```
SELECT kinase,ct, function
FROM {Prot} uniprot1:gene {gene1}, uniprot1:Gene {gene2},
     {gene2} uniprot1:locusName {kinase}, {Prot} uniprot1:annotation {annot1},
     uniprot2:Function_Annotation {annot2}, {annot2} rdfs:comment {function},
     {Kinase} ns0:Orf {orf}, {Kinase} ns0:Phos_count {ct}
WHERE gene1=gene2 and annot1=annot2 and kinase=orf and kinase="YBL105C"
USING NAMESPACE
uniprot1=urn:lsid:uniprot.org:ontology: ,
uniprot2=urn:lsid:uniprot.org:ontology: ,
rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns# ,
rdfs=http://www.w3.org/2000/01/rdf-schema# ,
ns0 = http://mcdb750.med.yale.edu/yeasthub/schema/schema14469.rdf
```

kinase	ct	function
"YBL105C"	"2"	"Required for cell growth and for the G2->M transition of the cell division cycle. Mediates a protein kinase cascade; it activates BCK1 which itself activates MKK1/MKK2."

Fig. 10. Syntax and output of example query 2.

3.2 Performance

Sesame allows a repository to be created using a database (e.g., MySQL), native disk, or main memory. We evaluate the performance of these approaches using example query 1 described previously. We run the same query twice against main memory, mySQL, and native disk repositories. Each repository stores the identical datasets with a total of ~800K triple statements.

Table 2 shows the amount the time (in milliseconds) it takes for query execution for each repository type. Both the main memory and MySQL approaches take about the same amount of time on the first query run (~300ms). On the second query run, the MySQL approach is 7 times faster than the main memory one due to a cache effect (the speed difference, however, is only a fraction of a second). The file-based approach takes the longest query execution time.

Table 2. Query performance.

Query run	Memory	MySQL	File
1	312 ms	308 ms	9929 ms
2	306 ms	44 ms	11045 ms

Table 3 shows the amount of time (in seconds) it takes to load an RDF-formatted UniProt data file, which contains yeast data only, into the three repositories. The file size is about 63 MB (~1.4 million triple statements). As shown in Table 3, the main memory approach has the best data loading performance, while the MySQL approach has the worst performance due to the overhead involved in creating data indexes. In conclusion, the main memory approach gives the best overall performance.

Table 3. UniProt data loading performance.

Load run	Memory	MySQL	File
1	38 s	651 s	262 s
2	40 s	646 s	275 s

3.3 Implementation

YeastHub is implemented using Sesame 1.1. We use Tomcat as the web server. The web interface is written using Java servlets. The tabular-to-RDF conversion is written using Java. To access and query the repository programmatically, we use Sesame's Sail API that is Java-based. We use MySQL as the database server (version 3.23.58) to store information about the correspondences between the resource properties and the query form fields. Such information facilitates automatic generation of query forms and query statements. We also use the database server to create an RDF repository for performance benchmark as described previously. YeastHub is currently running on a Dell PC server that has dual processors of 2 GHz, 2 GB main memory, and a total of 120 GB hard disk space. The computer operating system is Red Hat Enterprise Linux AS release 3 (Taroon Update 4).

4 DISCUSSION

Although the tab-delimited format is popularly used in distributing life sciences data, there are other data distribution formats such as the record format (or the attribute-value pair format), XML format, other proprietary formats. It would be logical to incorporate these formats into our RDF data conversion scheme. In the process of our RDF data conversion, we generate the corresponding RDF schemas. While our approach to generating new schemas allows existing properties that are defined in other schemas to be reused, there is a need to perform schema mapping at a later stage, as new standard RDF schemas will emerge. How to translate one RDF schema into another RDF schema would be an interesting semantic web research topic.

While URL's are commonly used as a means to identify resources on the web, they have the following problems.

1. The web server referenced by the URL may be broken or become unavailable. Also, when a new server replaces the old one, the URL may need to be changed.
2. The syntax of the URL may change over time as the underlying data retrieval program evolves. For example, parameter names may be changed and additional parameters may be required.
3. The data returned by a URL may change over time as the underlying database contents change. This creates a problem for researchers when they want to exactly reproduce any observations and experiments based on a data object.

To address these problems, the Life Science Identifier project (http://www-124.ibm.com/developerworks/oss/lsid/) has proposed a standard scheme to reference data resources. Every LSID consists of up to five parts: the Network Identifier (NID); the root DNS name of the issuing authority; the namespace chosen by the issuing authority;

the object id unique in that namespace; and finally an optional revision id for storing versioning information. For example, "urn:lsid:ncbi.nlm.nih.gov:pubmed:12571434" is an LSID that references a **PubMed** article. Each part is separated by a colon to make LSIDs easy to parse. The specific details of how to resolve the LSID to a given data object is left to an LSID issuing authority. In our case, we can potentially implement an LSID resolution server (or LSID issuing authority) for referencing data objects stored in our semantic web data warehouse.

To increase the performance of data querying and loading, we use the main memory approach to build the RDF repository. For large amounts of data, we may use the relational database or native disk repository for data archival purposes and load the datasets of interest from the archival repository into the main memory repository for speed performance. Also, if we have a computer cluster, a parallel main memory architecture may be used to allow multiple main memory repositories to be queried concurrently.

While RDF-based query languages are SQL-like, there are SQL features that have not been implemented in Sesame yet. For example, not all RDF query languages (e.g., RQL) support outer-join like queries. In other words, if any of the properties included in a join query have no values, all the corresponding triple statements will be omitted from the query results. To get around this problem, our RDF data format includes property tags that have no data values.

Also, it would be useful to implement the aggregate functions (e.g., sum and average using GROUP BY). Sesame currently does not support delete and update queries, although these operations can be performed using some programmatic graph interfaces. Another limitation is that Sesame does not have a way to identify the source of triples (statements) once they are loaded into the repository. This makes removal of triples from a repository difficult if the triples come from different RDF files and have overlapping namespaces. SPARQL is a new RQL standard addressing these issues (http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/).

To enhance the knowledge representation and inference capability of the semantic web, OWL (http://www.w3.org/TR/owl-xmlsyntax/), which is an extension of RDF, has emerged as an XML-based web ontology language. Support of reasoning using OWL is being incorporated into some RDF stores (e.g., Sesame and Tucana). This allows such RDF stores to transit from being data stores to becoming knowledge stores. There are questions (e.g., planning, explanation, and prediction) that cannot be answered by traditional database queries. However, they can be addressed by the kind of representation and reasoning provided by an ontological language such as OWL. This has been demonstrated in the context of reasoning about signaling network data (Baral et al. 2004). Our work also represents a step in this direction.

5 SUMMARY

We describe how to use a semantic web approach to facilitate data interoperability in the life sciences domain. We build a prototype data warehouse by using a native RDF database system (Sesame) to store and query diverse types of yeast genome data across multiple sources. Although we use yeast data as our demonstration, our data integration approach can be applied to other organisms.

In addition to using a RDF data store, we demonstrate how to use other RDF technologies including RSS and D2RQ to describe metadata and map data dynamically from a relational database to RDF format. Our system allows these RDF technologies to be tested and interoperated with each other. For example, we found a bug when using D2RQ to map an Oracle database to RDF, while it was working fine for MySQL. With the help of the developers of D2RQ, we were able to fix the bug to make D2RQ work for Oracle. It is worth noting that the semantic web and RDF are still relatively new technologies; as time passes and their use becomes more widespread, more efficient and robust triple stores will be developed and applications such as ours will benefit from this.

We introduce an RDF format into which tabular data can be converted. Our goal is to make it easy for life scientists to cooperatively publish and use their data in RDF format (especially if their data are already available in the tab-delimited format). The benefits of using RDF in life sciences applications include the following.

1. It standardizes data representation, manipulation, and integration using graph modeling methods.
2. It allows exploration of RDF technologies such as triple stores and RDF query languages to integrate a wide variety of biological data.
3. It facilitates development and utilization of standard ontologies to promote semantic interoperability between bioinformatic web services.
4. It fosters a fruitful collaboration between the Artificial Intelligence (AI) research community and the life sciences research community.

ACKNOWLEDGEMENTS

This research is supported in part by NIH grant K25 HG02378 from the National Human Genome Research Institute, by NIH grant T15 LM07056 from the National Library of Medicine, and by NSF grant DBI-0135442.

REFERENCES

Alfarano, C., C. E. Andrade, K. Anthony, N. Bahroos, M. Bajec, K. Bantoft, D. Betel, B. Bobechko, K. Boutilier, E. Burgess, K. Buzadzija, R. Cavero, C. D'Abreo, I. Donaldson, D. Dorairajoo, M. J. Dumontier, M. R. Dumontier, V. Earles, R. Farrall, H. Feldman, E.

Garderman, Y. Gong, R. Gonzaga, V. Grytsan, E. Gryz, V. Gu, E. Haldorsen, A. Halupa, R. Haw, A. Hrvojic, L. Hurrell, R. Isserlin, F. Jack, F. Juma, A. Khan, T. Kon, S. Konopinsky, V. Le, E. Lee, S. Ling, M. Magidin, J. Moniakis, J. Montojo, S. Moore, B. Muskat, I. Ng, J. P. Paraiso, B. Parker, G. Pintilie, R. Pirone, J. J. Salama, S. Sgro, T. Shan, Y. Shu, J. Siew, D. Skinner, K. Snyder, R. Stasiuk, D. Strumpf, B. Tuekam, S. Tao, Z. Wang, M. White, R. Willis, C. Wolting, S. Wong, A. Wrong, C. Xin, R. Yao, B. Yates, S. Zhang, K. Zheng, T. Pawson, B. F. F. Ouellette and C. W. V. Hogue (2005). "The Biomolecular Interaction Network Database and related tools 2005 update." Nucl. Acids Res. **33**(suppl_1): D418-424.

Apweiler, R., A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi and L.-S. L. Yeh (2004). "UniProt: the Universal Protein knowledgebase." Nucl. Acids Res. **32**(90001): D115-119.

Ashburner, M., C. Ball, J. Blake, D. Botstein, H. Butler, M. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, M. Harris, D. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. Matese, J. Richardson, M. Ringwald, G. Rubin and G. Sherlock (2000). "Gene ontology: tool for the unification of biology." Nature Genetics **25**: 25-29.

Baral, C., K. Chancellor, N. Tran, N. L. Tran, A. Joy and M. Berens (2004). "A knowledge based approach for representing and reasoning about signaling networks." Bioinformatics **20**(suppl_1): i15-22.

Cheung, K., D. Pan, A. Smith, M. Seringhaus, S. Douglas and M. Gerstein (2004). An XML-based approach to integrating heterogeneous yeast genome data. Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'04), Las Vegas, Nevada, IEEE.

Decker, S., S. Melnik, F. V. Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks (2000). "The Semantic web: the roles of XML and RDF." IEEE Internet Computing(Sep-Oct): 63-74.

Fenyo, D. (1999). "The Biopolymer Markup Language." Bioinformatics **15**(4): 339-40.

Goldbeck, J., G. Fragoso, F. Hartel, J. Hendler, B. Parsia and J. Oberthaler (2003). "The national cancer institute's thesaurus and ontology." Journal of Web Semantics **1**(1).

Guelzim, N., S. Bottani, P. Bourgine and F. Kepes (2002). "Topological and causal structure of the yeast transcriptional regulatory network." Nature Genetics **31**(1): 60-3.

Hanisch, D., R. Zimmer and T. Lengauer (2002). "ProML - the protein markup language for specification of protein sequences, structures and families." In Silico Biol. **2**(3): 313-24.

Hermjakob, H., L. Montecchi-Palazzi, G. Bader, J. Wojcik, L. Salwinski, A. Ceol, S. Moore, S. Orchard, U. Sarkans, C. v. Mering, B. Roechert, S. Poux, E. Jung, H. Mersch, P. Kersey, M. Lappe, Y. Li, R. Zeng, D. Rana, M. Nikolski, H. Husi, C. Brun, K. Shanker, S. G. N. Grant, C. Sander, P. Bork, W. Zhu, A. Pandey, A. Brazma, B. Jacq, M. Vidal, D. Sherman, P. Legrain, G. Cesareni, I. Xenarios, D. Eisenberg, B. Steipe, C. Hogue and R. Apweiler (2004). "The HUPO PSI's Molecular Interaction format—a community standard for the representation of protein interaction data." Nature Biotechnology **22**: 177-83.

Hucka, M., A. Finney, H. Sauro, H. Bolouri, J. Doyle, H. Kitano, A. Arkin, B. Bornstein, D. Bray, A. Cornish-Bowden, A. Cuellar, E. Dronov, E. Gilles, M. Ginkel, V. Gor, I. Goryanin, W. Hedley, T. Hodgman, J. Hofmeyr, P. Hunter, N. Juty, J. Kasberger, A. Kremling, U. Kummer, N. L. Novere, L. Loew, D. Lucio, P. Mendes, E. Minch, E. Mjolsness, Y. Nakayama, M. Nelson, P. Nielsen, T. Sakurada, J. Schaff, B. Shapiro, T. Shimizu, H. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner and J. Wang (2003). "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models." Bioinformatics **19**(4): 524-31.

Kumar, A., K.-H. Cheung, N. Tosches, P. Masiar, Y. Liu, P. Miller and M. Snyder (2002). "The TRIPLES database: a community resource for yeast molecular biology." Nucl. Acids. Res. **30**(1): 73-75.

Pedrioli, P. G. A., J. K. Eng, R. Hubley, M. Vogelzang, E. W. Deutsch, B. Raught, B. Pratt, E. Nilsson, R. H. Angeletti, R. Apweiler, K. Cheung, C. E. Costello, H. Hermjakob, S. Huang, R. K. J. Jr, E. Kapp, M. E. McComb, S. G. Oliver, G. Omenn, N. W. Paton, R. Simpson, R. Smith, C. F. Taylor, W. Zhu and R. Aebersold (2004). "A common open representation of mass spectrometry data and its application to proteomics research." Nature Biotechnology **22**: 1459 - 1466.

Spellman, P., M. Miller, J. Stewart, C. Troup, U. Sarkans, S. Chervitz, D. Bernhart, G. Sherlock, C. Ball, M. Lepage, M. Swiatek, W. Marks, J. Goncalves, S. Markel, D. Iordan, M. Shojatalab, A. Pizarro, J. White, R. Hubley, E. Deutsch, M. Senger, B. Aronow, A. Robinson, D. Bassett, C. Stoeckert and A. Brazman (2002). "Design and implementation of microarray gene expression markup language (MAGE-ML)." Genome Biology **3**(9): 1-9.

Wuchty, S. (2004). "Evolution and Topology in the Yeast Protein Interaction Network." Genome Res. **14**(7): 1310-1314.

Yu, H., D. Greenbaum, L. H. Xin, X. Zhu and M. Gerstein (2004). "Genomic analysis of essentiality within protein networks." Trends Genet. **20**(6): 227-31.