A Scale-Out RDF Molecule Store for Distributed Processing of Biomedical Data

Andrew Newman
School of ITEE
The University of Queensland
Queensland, Australia
+61733654044
anewman@itee.uq.edu.au

Jane Hunter
School of ITEE
The University of Queensland
Queensland, Australia
+61733651092
jane@itee.uq.edu.au

Yuan-Fang Li
School of ITEE
The University of Queensland
Queensland, Australia
+61733654044
liyf@itee.uq.edu.au

Chris Bouton
Computational Sciences Center of Emphasis
Pfizer, Cambridge, MA, USA
Christopher.M.Bouton@pfizer.com

Melissa Davis
Institute for Molecular Biosciences and
ARC Center of Excellence in Bioinformatics,
The University of Queensland, Australia
m.davis@imb.uq.edu.au

ABSTRACT
The computational analysis of protein-protein interaction and biomolecular pathway data paves the way to efficient *in silico* drug discovery and therapeutic target identification. However, relevant data sources are currently distributed across a wide range of disparate, large-scale, publicly-available databases and repositories and are described using a wide range of taxonomies and ontologies. Sophisticated integration, manipulation, processing and analysis of these datasets are required in order to reveal previously undiscovered interactions and pathways that will lead to the discovery of new drugs. The BioMANTA project focuses on utilizing Semantic Web technologies together with a scale-out architecture to tackle the above challenges and to provide efficient analysis, querying, and reasoning about protein-protein interaction data. This paper describes the initial results of the BioMANTA project. The fully-developed system will allow knowledge representation and processing that are not currently available in typical scale-out or Semantic Web databases. We present the design of the architecture, basic ontology and some implementation details that aim to provide efficient, scalable RDF storage and inferencing. The results of initial performance evaluation are also provided.

Categories and Subject Descriptors
I.2.4 [**Knowledge Representation Formalisms and Method**]: Representation Languages; J.3 [**Life and Medical Sciences**]: Biology and genetics

General Terms
Design, Experimentation, Standardization

Keywords
Semantic Web, protein-protein interaction, scale-out architecture, RDF molecules, integration

1. INTRODUCTION
BioMANTA is a collaborative project that aims to enable *in silico* drug discovery and development by identifying candidate therapeutic targets through the analysis of integrated datasets that relate molecular interactions and biochemical pathways with physiological effects such as compound toxicology and gene-disease associations. This requires intensive real-time analysis and feedback, across large disparate datasets in order to interactively explore the data space and identify lead candidates or new interaction networks.

Current protein-protein interaction data is distributed across a wide range of disparate, large-scale, publicly-available databases and repositories. The integration of the data in these datasets is required before researchers can perform complex querying and analyses over the data to reveal previously undetected pathways. Given the different naming conventions [16], different syntactic and semantic representations and descriptions and the massive scale of the datasets, precise and efficient integration is a very challenging problem. Current tools available for bioinformatics data integration and discovery vary widely in terms of quality, maintenance and applicability. There is a proliferation of many different tools for performing operations on many different kinds of data [29], but there is also a general lack of standards for representing data, and a slow uptake of existing data standards [16]. Consequently, the BioMANTA project is aiming to deliver a more standardised approach to the integration of PPI data, through a set of open source interoperable tools and datasets that can be re-used and applied to bioinformatics generally.

1.1 The Architecture
Existing RDF databases have typically suffered from limited scalability and poor or inefficient inferencing and querying[1]. While some stores offer a high level of scalability for a single node, there is little support for aggregation across multiple nodes. Inferencing is typically limited to either basic operations across large amounts of data or richer inferencing over small amounts of data – we require rich, complex inferencing over large amounts of data[2]. There are many problems associated with scientific data analysis including: algorithm intensity, nonlinearity and limitations on computer component bandwidth [17]. These issues pre-

WWW 2008, April 21--25, 2008, Beijing, China.

[1] http://esw.w3.org/topic/TripleStoreScalability

[2] http://esw.w3.org/topic/LargeTripleStores

vent interactive analysis over derived datasets. In order to overcome these difficulties, Gray recommended a number of mechanisms to expedite and improve scientific data analysis [17]:

- The use of standardised and precise metadata to describe the units, names, accuracy, provenance, capture details, etc., in order to help tools compare and process the data correctly;
- The creation and adoption of common terminologies using Semantic Web technologies (RDF and OWL);
- The use of set-oriented processing methods, such as Google's MapReduce [11].

So while the use of ontologies and other Semantic Web technologies such as RDF can provide the ability to integrate, reason and process over datasets, the magnitude of the processing required and the size of the datasets have prevented a speedy, efficient end-to-end solution.

In order to process data quickly, a parallel architecture-based technique known as MapReduce [11] is becoming increasingly popular. This data processing technique provides a common way to solve general processing problems and is closely aligned with the way data is acquired from experiments or simulations [17]. In a MapReduce system, a map function takes input key/value pairs and transforms them to output key/value pairs. The reduce function takes the values in each unique key and produces output values. MapReduce libraries are found in the majority of the most popular languages including Java, Javascript, C++, Perl, C#, Python, Ruby, and Scala. The advantages of this architecture are numerous [11, 45] and include:

- A programming model that is abstract, simple, highly parallel, powerful, easy to maintain, and easy to learn;
- An ability to efficiently leverage low-end commodity hardware;
- Easy deployment across hundreds to thousands of nodes on internal or external hosting services; and
- Robustness and ability to recover from data corruption or the loss of individual nodes.

Our hypothesis is that Semantic Web applications can benefit from the adoption of a scale-out architecture together with MapReduce data processing, in order to speed up querying, inferencing and processing over large RDF triple stores of scientific and biomedical data.

1.2 Data and Ontologies

The primary aim of the BioMANTA project is to integrate data from protein datasets such as MPact [19], DIP [39], IntAct [26] and MINT [9] using a common model for proteins and protein-protein interaction data to enable data harmonisation. The common model is represented as a BioMANTA OWL-DL ontology[3] that we have developed. This ontology was developed by reusing vocabularies from well-established ontologies such as Gene Ontology [1], Cell Type ontology [2], BioPAX [4], PSI-MI [21], and others such as NCBI taxonomy. Based on the BioMANTA ontology, protein datasets are converted to RDF instances and stored in a distributed RDF triple store where they are available for subsequent analysis and querying.

The heterogeneity of naming conventions across the disparate datasets is a major problem. In addition, each dataset has its own method for protein identification. There have been previous attempts at naming standardization but they have had limited effect [16]. We believe that inventing another naming convention or trying to reach a consensus will not solve the identification problem. Instead, we have developed an identity reconciliation process. Firstly, RDF blank nodes are used to represent the "real" proteins and provide the hub that links to the relevant entries in different (translated) datasets to create a single representation encompassing all information about a particular protein. This approach enables all three levels of "attitudes" of knowledge representation (record, statement and domain) [38] pertaining to a particular protein, to be incorporated in the RDF document, enabling highly sophisticated, multi-level queries to be expressed.

In a scale-out architecture large RDF documents (graphs) need to be sub-divided into smaller ones for distributed processing. During querying, processing results must be merged in order to eliminate duplicate protein representations. This requires the disambiguation and identification of blank nodes – which is not a simple process. As each blank node's scope exists only within the enclosing document, blank nodes are not *globally* addressable. Hence, a consistent way of uniquely referring to them is required. The concept of RDF molecules [12] was proposed to tackle the problem of addressing blank nodes by decomposing an RDF graph losslessly into a set of molecules which distributes updates to graphs. In the work described here, we extend the definition of RDF molecules to make the storage, retrieval and querying more efficient in the distributed environment.

In the remainder of this paper we describe the novel approach to semantic integration of biomolecular data that we have implemented and evaluated. This approach combines the scalability and performance of a scale-out architecture, with the simple and efficient MapReduce programming model and Semantic Web technologies to enable interactive querying, inferencing, analysis and modelling of disparate protein-protein interaction and pathway datasets. In Section 2, we describe related works. Section 3 describes the proposed application and the high-level architecture of the system. In Section 4 we present the ontologies we have developed, the datasets we have integrated to date and the types of queries we are aiming to support. In Section 5 we describe the extended RDF molecule approach we have implemented. Section 6 presents an initial performance evaluation for decomposing RDF files and the integration of proteins Section 7 concludes with a summary of the outcomes to date and future work plans.

2. Related Works

2.1 The Scale-Out Architecture

For a relatively new architecture, scale-out MapReduce systems have already received very promising and positive feedback and evaluation results. Benefits include: better price/performance, successful application to many different domains, and open source implementations.

Google's initial work using these MapReduce scale-out techniques has included: indexing the web, statistical analysis of Web site usage, general data storage and querying, map and satellite imagery processing, and social networking [8]. Similarly, Yahoo has been applying the technology for: "search and information retrieval, machine learning and data mining, microeconomics, community systems and media experience and design" [43]. Other successful applications include: indexing and searching web documents [27], natural language processing [33], learning algorithms for multicore systems [10] and simulation [28].

The Hadoop[4] project provides an open source implementation of Google's scale-out MapReduce, system including the Hadoop Distribute File System (HDFS), MapReduce and HBase (a Big-Table clone).

[3] http://biomanta.sourceforge.net/2007/07/biomanta_extension_02.owl

[4] http://hadoop.apache.org/

While there are no known publications that describe the use of MapReduce scale-out architectures to store and process RDF, there have been initial implementations and research into similar, overlapping areas including: RDF stores using "shared nothing" clustering, extending MapReduce higher-level operations, and column databases for storing and querying RDF.

The YARS2 federated RDF repository and the SWSE (Semantic Web Search Engine) architecture use a "shared nothing" approach to achieve scalability [20]. This has some conceptual similarities to our data acquisition architecture. However, it is still bound to indexing and querying, and does not share the attributes of a MapReduce scale-out solution with its ability to perform arbitrary processing and indexing schemes.

The design of BigTable and HBase is similar to column databases such as Sybase IQ, LucidDB, Metakit, KDB, C-Store [42] and Monet [5]. These databases were specifically designed to get the best performance from modern hardware architecture. There is also some initial research currently underway investigating the use of C-Store and MonetDB [31] for storing and querying RDF data as well as using these databases to handle scientific data [22]. Our approach differs from these approaches in a number of ways:

- We create a generic store for triples of any predicate, rather than creating one table per predicate;
- Our clustered approach differs substantially from their database architecture;
- We do not support ACID (Atomicity, Consistency, Isolation and Durability) database transactions;
- Column databases do not have a MapReduce-like processing framework, and don't combine processing and data management in the same way.

To the best of our knowledge, the work described in this paper represents the first attempt to apply a scale-out distributed computing approach to expedite the querying and processing of data in a large scale-out RDF triple store. Although we specifically apply it to protein-protein interaction (PPI) data, there are undoubtedly many other suitable applications that require the integration and processing of large scale distributed datasets (e.g. climatology, geosciences, astronomy).

2.2 RDF Molecules

The concept of RDF molecules was first proposed in [12] as a method that provides the optimum level of granularity between RDF graphs and triples. Given an RDF graph *G*, the set of molecules are the smallest sets of triples into which *G* can be decomposed without loss of information. Figure 1 from [12] shows the different granularity levels of various RDF constructs. This illustrates that RDF molecules sit between named graphs [23] and triples in terms of granularity.

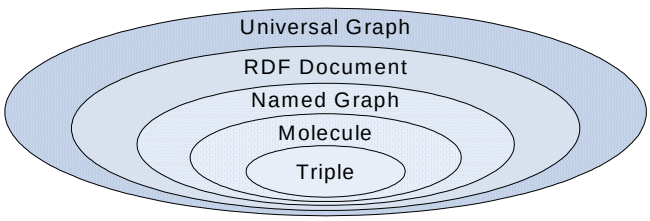

Figure 1 - Relative Granularity Levels of RDF Constructs.

Formally, given an RDF graph *G* and a background ontology *W*, a pair of operators *(d, m)* is defined for decomposition and merging.

$$M = d(G, W)$$
$$G = m(M, W)$$

Where, *M* is the set of molecules as the result of decomposition of *G* with regards to *W* using decomposition operator *d*. The merging operator *m* merges *M* back to the same graph *G*, also with respect to the background ontology *W*. The set of molecules *M* are mutually independent in the sense that no blank node is shared among them. Hence, they can be individually processed and later merged to construct the RDF graph *G* losslessly.

Three types of RDF nodes were defined based on their groundedness. A node is:

1. Naturally grounded if it is either a URI reference or a literal;
2. Functionally grounded if it is a blank node and is uniquely determined by a functional/inverse functional property and another node;
3. Contextually grounded if it is a blank node and not functionally grounded.

There are also three types of molecules, based on the types of nodes they contain:

1. Terminal molecules contain only grounded nodes and/or functionally grounded blank nodes, which are all "closed".
2. Non-terminal molecules contain one "open" blank node.
3. Contextual molecules contain at least one contextually grounded blank node.

Two types of decomposition were defined: naïve decomposition, in which no background ontology is consulted; and functional decomposition, in which an OWL ontology is queried for functional dependency between nodes.

A number of extensions to RDF similar to RDF molecules have also been proposed in the past. **Named graphs** [23] are an extension of RDF that enables the specification of an RDF graph through a set of RDF statements. The division of statements into sub-graphs is arbitrary. The ontology author is responsible for manually constructing the graphs and naming them. Hence no automated process is available.

Concise Bounded Description (CBD) [41] is defined as a sub graph of statements about a particular resource. Given a particular resource, its CBD contains:

1. All statements in the source graph in which the subject of the statement is the particular resource;
2. Recursively, for all statements identified in the subgraph thus far having a blank node object, all statements in the source graph where the subject of the statement is the blank node in question and which are not already included in the subgraph.
3. Recursively, for all statements included in the subgraph thus far, for all reifications of each statement in the source graph, the CBD beginning from the rdf:Statement node of each reification.

A drawback of CBD is that it only looks at subject nodes in RDF triples. Hence, the CBD created for a resource node may not include all the information.

Minimum Self-contained Graphs (MSG) [15] is a proposal for the decomposition of an RDF graph into self-contained subgraphs. Given an RDF statement, its corresponding MSG includes (a) the statement itself and recursively, and (b) for all the blank nodes *involved* in the MSG so far, all the statements of MSGs involving these blank nodes. Compared to CBD, MSG looks for statements to be included in the MSG in both directions. Hence, it results in a lossless decomposition. One potential drawback of MSG occurs when the graphs contain many blank nodes. In this situation, the resulting subgraphs may be very large.

Based on the above comparison, RDF molecules provide the best granularity as they offer automated, unambiguous and lossless decomposition, whilst not suffering from the potential scale problem that MSGs face.

The REDD algorithm [13] was defined to optimize RDF storage by reducing redundancy caused by blank nodes. It finds redundant blank nodes by repeatedly identifying "connected subgraphs", constructing and executing queries on these subgraphs. The complexity of the REDD algorithm is $O(n^2)$, where n is the number of triples in the graph. Creating a chain of triples (a connected subgraph) by navigating predicates was suggested as part of the REDD algorithm. It was also suggested that a triple ordering be created so that inclusion of other connected subgraphs or supergraphs be determined, apart from simple equality. These concepts are also included in our extension of RDF Molecules.

3. The BioMANTA Project

3.1 The Biomolecular Application

The first step in the BioMANTA project was to identify the relevant available datasets that we wanted to integrate, and to develop the common model or ontology that we would use to do so. Candidate databases included BIND, DIP, HPRD, KEGG, Reactome, Ingenuity and GeneGo. Due to the intrinsic representation of semantic data as a network of triples, protein-protein interaction data is highly amenable to representation in Semantic Web form.

While it might be assumed that an interaction could be represented simply as a single triple (e.g. a protein-protein interaction: <proteinA> <interactsWith> <proteinB>) much more information must be included to capture biologically-relevant aspects of interactions. A protein, for example, has a number of attributes that must be included in its representation. These include simple attributes that can be used to identify proteins, such as "full name", "short name", "sequence", "species" and "gene symbol", as well as any number of "synonyms" and "accession numbers", which may be associated with the protein in various resources. To support these identifying attributes, a model of a protein would require a minimum of seven triples. Interactions also require multiple triples for accurate representation. Key concepts in the representation of interactions include "participants", "interaction type", "interaction detection method", and "references". Many attributes are composed of multiple attributes; for example, a "reference" may have an associated "PubMed Identifier" and "abstract", which in turn may have a "PubMed URL". An example of this structure is shown in Figure 2.

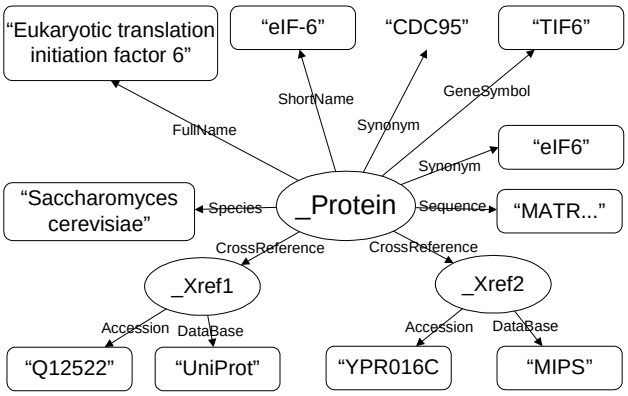

Figure 2 - A protein with relevant information represented as triples.

A number of public efforts have used RDF and OWL to represent many types of biological information, including protein interaction data. The UniProt [44] database was recently migrated into a more semantic form. YeastHub [25] also uses Semantic Web technologies to organize and integrate biological data sources. The Semantic Web Applications in Neuromedicine (SWAN) [14] project is utilizing semantic technologies to enable collaborative research. The BioPAX initiative is developing a data format for use in the modeling of numerous levels of biological pathway information. The Reactome project [24] has provided its data in the BioPAX Level 2 format [4].

We chose to use OWL DL to construct a high-level ontology to integrate concepts from relevant biological ontologies and vocabularies. We reviewed ontologies listed by the Open Biomedical Ontology Foundry (OBO Foundry) [40] and the National Center for Biomedical Ontology (NCBO) [35]. Of the approximately 70 ontologies listed at these sites, around three quarters are written using the OBO format, with the remainder using formats including OWL, Protégé and plain text. Some of these, such as BioPAX and the Protein Standards Initiative Molecular Interaction vocabulary (PSI-MI) [21], provide significant coverage over concepts relevant to the domain. Others, such as the Gene Ontology (GO), Sequence Ontology and the NCBI-taxonomy only intersect with the field.

In particular, merged components from two ontologies were used to describe molecular interactions in our ontology - the OWL ontology BioPAX and the OBO ontology PSI-MI. BioPAX is designed to describe pathway rather than specific molecular interaction data. However, of the approximately 40 classes and 70 properties that BioPAX defined, many are key concepts and relationships necessary to describe molecular interactions. The PSI-MI vocabulary on the other hand, is specifically designed to describe molecular interaction data and captures >800 concepts from the domain. However, it is represented in OBO and expresses only hierarchical relationships between classes. Hence, we adopted and merged components from both ontologies during the development of the BioMANTA ontology.

Four protein interaction databases were initially selected for conversion to RDF. They are MPact, DIP, IntAct, and MINT. These resources were selected for several reasons: they are participants in the International Molecular Exchange (IMEX) Consortium[5], which makes data available under the Creative Commons license arrangements[6]; they contain human-curated interaction data which is better quality than automatically mined or predicted interaction data, and these datasets are available for download in a standard data exchange format, the Protein Standards Initiative (PSI) Molecular Interaction (MI) XML format [21]. PSI-MI provides a hierarchical controlled vocabulary for recording information about molecular interactions. While this format is suitable for the exchange of data between resources, it lacks the ability of description frameworks like RDF and OWL to support Semantic Web applications. These databases also contain a very large number of proteins and interaction instances. In excess of six million triples are required to express the interaction data contained in these resources (Table 1).

A significant saving in triple numbers may be achieved by grouping proteins or interactions by some common property that can then be represented in the metadata of the document containing that set. For example, provenance information such as the source database of interactions, version information, data acquisition dates and license information can be recorded as document metadata, thus removing the necessity to record this information for each interaction to which it pertains. Likewise, if proteins are grouped according to species, then this attribute is represented in the graph metadata for all proteins contained in the graph, remov-

[5] http://imex.sourceforge.net/

[6] http://creativecommons.org/

ing many thousands of triples from the instance data. Even with the shift of significant amounts of information into graph metadata, a large number of triples are required to represent each interaction, regardless of the ontology selected. We have conservatively estimated that each protein requires ten triples, while each interaction requires fifteen. While these numbers may seem small, when considering the volume of interaction information stored in publicly available resources, it becomes obvious that the resulting RDF data sets for molecular interactions is in the order of millions of triples. This number is fluid, as it depends on the information being modelled and the techniques used to express them.

Table 1 - Estimated IMEX Protein-Protein Data Sizes.

Database	MPact	DIP	IntAct	MINT
Number of proteins	5028	19490	57360	28186
Number of binary interactions	16504	56186	152100	103808
Estimated protein triples	50280	194900	573600	281860
Estimated interaction triples	247560	842790	2281500	557120
Total per DB	297840	1037690	2855100	1838980

Once this data is represented in RDF, inferencing is used to infer additional information that is the logical consequence of the given statements. For example, if two proteins are from the same species, and have the same protein sequence, but have different names, it may be inferred that those names are synonyms. Likewise, if an interaction is purported to occur between proteins from different species (for example, a human and a hamster) it may be inferred that this interaction will never occur in nature, and is thus likely to be a false positive interaction.

3.2 The Architecture

The BioMANTA system requires the use of both online and batch processing. The online processes are performed against existing indices, and allow users to quickly query against the pre-computed datasets. The batch processing of queries supports more complicated and potentially non-terminating processes. Figure 3 shows the general processing that occurs within the BioMANTA system. The instance data and ontologies (either pre-processed or generated by crawling data sources on the Web using Nutch) are converted to RDF that conforms with the BioMANTA ontology. Pellet is then used to infer new statements directly related to the instances being added. RDF/XML is generated and then read in by the cluster into the HBase store. The RDF graph is broken down into smaller, sub-graphs and then added. HBase then indexes and stores the data across Hadoop's Distributed File System. The query engine processes the queries and returns the immediate results, which may indicate that further processing is required. If further processing is required, the user is notified and the additional data transformation jobs are added to process the data.

Examples of simple queries include: join and optional (left outer join) queries, Lucene text searching including RDF literals, pre-computed inferred statements, pre-computed network queries such as protein to protein interactions, and identifier mapping. Examples of batched operations include: shortest path queries, ad-hoc computing on interaction networks, and probabilistic queries. For the current purposes of determining the quality of the interactions integrated, we perform queries such as: the number of interactions with specific proteins, interactions that occur with certain types of observations, the number of experimental methods and for every PubMeD ID how many interactions are recorded.

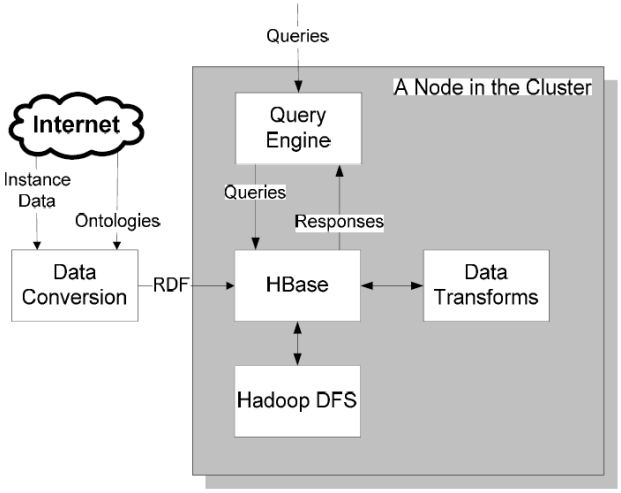

Figure 3 - BioMANTA Architecture.

4. A Uniform Representation of Biomedical Data in OWL

4.1 The BioMANTA Ontology

The BioMANTA ontology is an OWL DL ontology that builds on existing biological ontologies, such as BioPAX Level 2 [4], Cell Type ontology [2] Gene Ontology [1], and NCBI Taxonomy[7] , to enable the integration of the various facets of knowledge about protein-protein interactions. The ontology combines the use of top-down and bottom-up development taking terms as appropriate in order to leverage existing datasets. We have reused vocabularies where appropriate such as the Cell Type ontology (a structured controlled vocabulary[8] taken from OBO Foundry that was converted to OWL format[9]). Figure 4 shows the major components of the BioMANTA ontology.

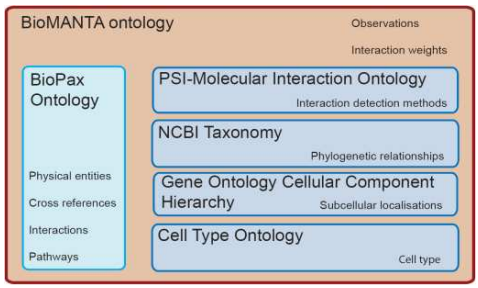

Figure 4 - Major Components of the BioMANTA ontology.

Our approach to knowledge representation combines two of the three levels of attitudes to data modeling: record and statement [37]. Our ontology allows us to express: "there exists a protein (the record level) by asserting its existence, and database A says it has these properties and database B says it has these properties"

[7] http://www.ncbi.nlm.nih.gov/Taxonomy/

[8] http://obofoundry.org/cgi-bin/detail.cgi?id=cell

[9] http://biomanta.sourceforge.net/2007/07/celltype_instance_edit.owl

(the statement level) by recording provenance information. We also align the experimental provenance information (statement level) to improve query quality – allowing us to filter out various experimental types. This work is based on object identification or record linkage [3] which seeks to integrate various data sets across databases and the Semantic Web [18]. In our current modeling approach we do not capture knowledge on the domain level yet. However, domain level modeling may be incorporated at a later stage for more complex reasoning.

In the ontology, the hierarchical but expressive properties of PSI-MI, such as those used to record participants and experimental methods of interactions, are combined with the extensibility and richer relationships available in BioPAX (as it is expressed in OWL). Among others, the ontology includes definitions of the following key concepts:

- Observation types including: Experimental, Predicted and Inferred,
- Provenance information including: data source, the type of experiment, the cell type, inferencing method, sub-cellular location and observation reference (a BioPAX publication cross reference).

While BioPAX is expressed in OWL there are numerous problematic issues associated with its modeling technique [37] [36] [30]. One of the most detrimental to our requirements is the lack of context or meaning when the `openControlledVocabulary` class is used to include links to external terms. To overcome this, we developed a process of taking converted OBO to OWL ontologies and defining classes and instances to represent these richer relationships.

By mapping and linking to existing component ontologies using OWL object properties, the BioMANTA ontology serves as a well-structured model for representing RDF instances. These RDF instances represent an integrated view of individual proteins and all of the information about them. The next subsection provides a brief account of the integration process.

4.2 The Integration Process and Datasets

Various protein datasets such as DIP, IntAct, MPact, and UniProt [44] often contain partially overlapping information about proteins and genes. The integration of the data in these datasets provides a uniform representation of a significant proportion of the available information pertaining to protein-protein interactions, allowing for flexible querying on topics such as tissue/organ expressions, species, genomic sequences, developmental states, etc.

There are two major challenges that must be overcome to perform this integration. Firstly, different datasets often use different naming conventions; hence, it is difficult but important to be able to identify the "same" protein in different datasets. For example, the protein identified as "27628" in DIP is the same protein as the one identified as "115 dax human" in IntAct. Both identifiers are maintained, to allow querying and retrieval of associated data and properties from both databases. Secondly, as datasets sometimes contain omissions, duplication, inconsistency and noise, it is not advisable to rely on the matching of names as an indication of the same protein. Given the above challenges, we decided to use a combination of UniProt IDs and genomic sequences to distinguish and identify proteins. This helps to ensure the high quality of the resultant data. The integration process can be conceptually described in the following steps.

1. **PSI-MI to RDF translation** - The XML datasets in PSI-MI [21] format are translated to RDF. This involves modeling all of the information associated with proteins and interactions using the RDF constructs, concepts and properties defined in the BioMANTA ontology. For example, the organism, the local identifiers and the genomic sequence are extracted and associated with each protein.
2. **UniProt ID augmentation** – Because UniProt is a comprehensive protein database, we decided to use UniProt IDs in the integration process to merge proteins from different datasets. However, not all datasets contain UniProt IDs. In this step, with the help of external mapping files [32] between local MPact IDs (CYGD IDs)[10] and UniProt IDs, a UniProt ID is added to each protein instance.
3. **Sequence augmentation –** Proteins that have genomic sequences can be disambiguated by evaluating the sequence, using tools such as Blast. Equivalent sequences are then used to identify equivalent proteins across data sources. In this step, all the missing sequences of proteins are added to the RDF instances from external mapping files for individual datasets.
4. **Protein integration** - the final step involves identifying equivalent proteins in different datasets (by matching UniProt IDs and sequences) and merging them into a single, uniform representation. Proteins with different UniProt IDs are considered to be different; those with same UniProt IDs but different sequences result in warnings; proteins with matching IDs and sequences will be merged into a single protein, together with their annotations.

Instead of creating another naming convention we use blank nodes to represent proteins and add properties to them (including provenance information). Blank nodes are also used to represent interactions, cross references, experimental methods, etc., for the benefits of time and cost savings [18]. Figure 2 above depicts such a merged protein. The next section details our approach to the efficient processing of RDF graphs with blank nodes.

5. Extended Molecules

In order to incorporate as many datasets as possible without restriction, we decided to allow blank nodes in RDF documents. This decision presents a challenge for the distributed processing of RDF documents as blank nodes are only addressable locally within a document. RDF molecules [12] provide a mechanism for decomposing an RDF graph into a set of self-contained molecules, each of which contains all (transitively) connected blank nodes. This enables an RDF graph to be losslessly decomposed, distributed for processing and subsequently merged, as depicted in Figure 5 below.

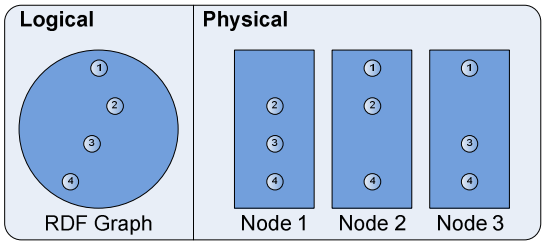

Figure 5 – An RDF Graph as molecules across multiple nodes in a computing cluster.

In order for an RDF molecule store to support efficient storage and querying of large-scale RDF documents, it is important that basic operations such as decomposition and merging on molecules is efficient. Hence we extend the original definition of molecules in [12] to include (1) hierarchy and (2) ordering.

[10] ftp://ftpmips.gsf.de/yeast/EBI/

5.1 Hierarchies

In the original definition, molecules are flat and each molecule contains a set of RDF triples. We believe having hierarchical molecules helps to better reflect the structure of the underlying RDF document. These extensions to molecules accurately reflect a structure found in biological and other data and represent relationships found in databases similar to where one relation refers to another via a foreign key.

Another important reason for adding hierarchies is to be able to identify equivalent blank nodes based on *context* instead of on internal identifiers. Given the same context we can determine blank node equivalence and remove redundant information.

Figure 6 below shows a simplified observed protein-protein interaction in N3 format.

```
{_:1 observedInteraction _:2}
{_:1 type ExperimentalObservation
{_:2 participant _:3}
{_:3 hasUniprotID ‘p32379’}
{_:2 participant _:4}
{_:4 hasUniprotID ‘p46949’}
```

Figure 6 A Protein-Protein Interaction in RDF.

The nested structure of the above RDF fragment cannot be represented by the original molecules since it will be a set containing all the triples. By allowing molecules to contain submolecules, it is possible to represent the structure. The above graph is decomposed into a molecule as shown in Figure 7.

The different indentation levels signify different nesting levels or submolecules. The “root triples” of a molecule are the triples in the first level of the molecule. A “head triple” is the first triple, as they are ordered, in the set of root triples. A “linking triple” is a triple that has two blank nodes, the second blank node linking to a submolecule.

```
{ _:1 type ExperimentalObservation }
{ _:1 observedInteraction _:2 }
      { _:2 participant _:3 }
            { _:3 hasUniprotID ‘p32379’ }
      { _:2 participant _:4 }
            { _:4 hasUniprotID ‘p46949’’ }
```

Figure 7 The Hierarchical Molecule Corresponding to the Triples in Figure 6.

Figure 7 shows a molecule with three levels, two root triples (the first two) and the head triple is “_:1 type ExperimentalObservation”.

5.2 Ordering

The other major extension to molecules that we implemented is *ordering*. Maintaining ordering is important for the efficient comparison of molecules and triples for graph and molecule merging.

The ordering is defined over triples given a molecule. The “less than” relationship between two triples is based on the comparison between their subjects, predicates and objects.

For two nodes, the ordering is determined by the following rules.

- Node type:
  - Blank node type, which is less than;
  - URI reference node type, which is less than;
  - Literal node type.
- Node value
  - Comparison of string value of the nodes

The ordering of two triples is based on the comparison of their nodes in turn. If subject nodes are equal, predicate nodes are compared. If predicate nodes are equal, then the object nodes must be compared.

The comparison of two molecules is based on the head triples they have. For molecules $molecule_1$ and $molecule_2$ and their head triples t_1 and t_2, $molecule_1 \oplus molecule_2$ iff $t_1 \oplus t_2$, where the symbol $\oplus$ represents $<$, $=$ or $>$.

5.3 Algorithms

In this section, we present algorithms for molecule-related operations such as naïve graph decomposition (no background ontology) and molecule merging. There are a number of advantages associated with this approach compared to the functional approach:

- Less duplication across molecules – the functional decomposition will generally result in blank nodes shared across multiple molecules whereas naïve decomposition will generate one molecule containing all such blank nodes.
- As the decomposition and processing does not need to consult an ontology, it is generally faster and is easier to implement.

As described in [12], the naïve graph decomposition algorithm decomposes a graph into a set of molecules. The decomposition of a local RDF graph into a set of molecules is described in the pseudocode shown below. We rely on the equality of the blank node identifiers (a combination of a UUID and a surrogate numeric identifier) when decomposing triples from a local graph.

```
AT is the set of added triples (initially empty).
LGT is a sorted set in descending order (defined above) of triples from a local graph.
FOR EACH Triple T from LGT not in AT
      Create a new molecule M adding T.
      IF T is Grounded THEN
          Add T to AT.
      ELSE
          findEnclosedTriples(M).
      END IF
END FOR
findEnclosedTriples(M)
      T is the HeadTriple of M.
      BTS is a set of all triples which contain T's blank nodes.
      FOR EACH Triple BT from BTS not in AT
          Create a new molecule SM adding BT.
          Add BT to AT.
          findEnclosedTriples(SM)
          IF BT is a Link Triple THEN
              IF BT's object node equals M's subject node THEN
                  Add M to SM.
                  SM becomes M.
              ELSE
                  Add SM to M.
              END IF
          ELSE
              Add BT to M.
          END IF
      END FOR
      Add all triples found to the set AT.
END findEnclosedTriples
```

There are three cases to consider when identifying sub-molecules:

- If the head triple is a link triple and the triple to add has a subject that is equal to its object then the triple is added to the head triple.
- If the identified sub-molecule contains a triple which links to the head of the current molecule then the current molecule is added to the sub-molecule and the molecule used from then on is the sub-molecule. In other words, the contents of the molecule are added to the sub-molecule which becomes the molecule used in future operations.
- If the identified sub-molecule does not contain a triple which links to the current molecule then it is added to the current molecule.

The complexity of the above graph decomposition algorithm can be analyzed as follows. Assume that all basic operations such as

adding one triple to a molecule, comparison between two nodes, getting the subject/object node from a triple; testing whether a triple is a blank node, and creating a molecule, etc., all take constant time *O(1)*. The complexity of the algorithm depends on the number of blank nodes of the graph being decomposed. For example, suppose we have a graph *G* with *n* triples:

- The best case is when no triple contains blank nodes. In this case, both the subject and object nodes of each triple are tested for blank node. The triples are subsequently added to a new molecule. Four constant-time operations are performed for *n* triples. Hence, the complexity is linear to the size of the graph *O(n)*.
- The worst case is when all triples share, recursively, some blank nodes and they end up in one molecule with n levels (one triple at a level). In this case, the molecule is a chain of triples. As a triple is only added to a (sub) molecule once, it is only compared to the head triple of the enclosing molecule once. Hence, only a constant number of basic operations are performed for adding each triple. Hence, the time complexity is still *O(n)*.

Therefore, the complexity of the decomposition algorithm is O(n), linear to the size of the graph. Also note that three indices are maintained for subject (s), predicate (p) and object (o): (s p o), (p o s) and (o s p), where all the triples in the graph are stored in all three indices. By storing these indices in hash maps, the retrieval of triples takes constant time.

The merging of molecules depends on the presence of a one-to-one correspondence between blank nodes. Next we present the algorithm for finding the mapping between molecules m1 and m2, shown below.

```
findBlankNodeMap(m1, m2)
   BM is a map of blank nodes from m1 to m2 (initially empty).
   FOR EACH root triple t1 in m1
      Find the root triple t2 from m2 that corresponds to t1.
      LET sm1 = m1.submolecule for t1.
      LET sm2 = m2.submolecule for t2.
      IF sm1 != null AND sm2 != null THEN
         nm = findBlankNodeMap(sm1, sm2).
         IF nm = empty THEN
            return empty map.
         ELSE
            add nm to BM.
         END IF
      ELSE IF t1.submolecule = null AND t2.submolecule = null THEN
         add map between blank nodes in t1 and t2.
      ELSE
         return empty map.
      END IF
   END FOR
   return BM.
END findBlankNodeMap
```

For each root triple, get the sub-molecules of m1 and compare them to the triples of m2. If the two triples are equal (using the blank node ID), then the corresponding blank nodes of the two triples are added to the map. This process stops when all levels of one molecule have been considered.

The complexity of the findBlankNodeMap algorithm depends on the number of comparisons between triples of the two molecules. Note that having hierarchies helps to greatly reduce the number of comparisons as comparisons are only made for sub-molecules on the same level.

Without loss of generality, let us assume that m1 has fewer levels of submolecules. Let the number of levels of m1 be *m*, and the number of triples on level *i* be n^1_i. For the first *m* levels, let the number of triples of molecule m2 be n^2_i. Thus the complexity of the findBlankNodeMap algorithm is:

$$C_2^1 = n_1^1 * n_1^2 + \cdots + n_m^1 * n_m^2 = \sum_{i=1}^{m} n_i^1 * n_i^2$$

The merging algorithm for the original molecule definition would require the comparison of all proteins, resulting in complexity of $\sum_{i=1}^{m} n_i^1 * \sum_{i=1}^{m} n_i^2$, which is strictly larger than the above complexity result and the difference is greater with the increase in the number of levels.

The extended molecules is an important component of the BioMANTA project, together with the scale-out architecture, the molecule store will enable efficient storage, retrieval, querying and analysis of integrated biomolecular data. In the next section, we give a brief account on the performance evaluation of molecule-related algorithms and the integration process.

6. Evaluation

We have chosen to evaluate two separate but related tasks in regards to our implementation: domain-independent graph isomorphism and domain-specific dataset merging. The evaluation seeks to remove as many redundancies between two graphs/datasets as possible. Our implementation is Java based (JRDF[11]), therefore we have concentrated on Java RDF implementations in order to offer the best comparisons. JRDF provides two RDF models one using a typical approach used by Kowari, Sesame, Jena and YARS (local graph) and a molecule based approach (global graph). Jena is used in the first evaluation as it implements graph isomorphism and Sesame is used for the second as it most closely aligns with our implementation architecturally.

The first evaluation seeks to identify equivalence between RDF graphs. This is a significant barrier to efficient data integration - equivalence relationships between corresponding nodes in two graphs needs to be established in order to integrate them. It has been shown in [7] that RDF graph equality is equivalent to the problem of graph isomorphism [34]. Two graphs are isomorphic if there is a one-to-one correspondence between the sets of nodes of the two graphs.The algorithms described in the previous section have been used to identify graph isomorphism. In order for us to store RDF molecules in JRDF a local graph is created from initial data sets, the graphs are decomposed into molecules and these are then merged to remove redundancy statements. We compare this RDF molecule approach with Jena which uses an algorithm that classifies nodes into classes according to their connectedness with other nodes. An exhaustive matching of nodes between equivalent classes is then performed. The algorithm used by Jena has a worst case complexity that is exponential - whereas our algorithm is $O(n^2)$.The current weakness in our algorithm is that it must decompose a fully grounded graph, whereas Jena's algorithm avoids doing this. The algorithm can handle certain triple structures better than Jena including looping blank nodes and chained blank nodes. Chained blank nodes takes the form _:1 p _:2, _:2 p _:3 and so on whereas looping blank nodes are a chain with the final triple's object pointing back to the first triple's subject (e.g., _:1 p _:2, _:2 p _:1). Our algorithm can decompose and remove 500 redundant chained blanks nodes (with a depth of 20) in 74.9 seconds compared to Jena's 325.8 seconds. At smaller depths this advantage reduces to only twice as fast at a depth of 10 and approximate parity at a depth of 3 and smaller.

As described in Section 4.2, the integration process is an important component of the BioMANTA project. We evaluated the performance of the integration process as implemented using Sesame 2.0 [6] and JRDF.

[11] http://jrdf.sourceforge.net/

Furthermore, we compare the performance of two (translated) RDF documents about yeast in the IntAct database, **yeast_small-01** and **yeast_small-03** [12]. Brief statistical data for the two RDF documents and the merged document is given in the table below. Note that out of the 1395 proteins, 85 are merged because they have the same UniProit IDs and genomic sequence strings.

Table 2 - Statistics from Merging Yeast Data

	Yeast_small-01	Yeast_small-03	Merged
No. triples	27582	50267	70643
No. proteins	503	893	1395

These tests were conducted using local RDF graphs - ones that use a nodepool/value store which creates unique identifiers (localization). The yeast data are integrated using two different approaches: BioMANTA/JRDF and Sesame. The time taken to perform the integration (in seconds) is given in the table below and visualized in the Figure 8.

Table 3 - Relative Performance of BioMANTA and Sesame Yeast Data Integration (time in seconds)

	BioMANTA/JRDF	Sesame
Reading RDF	14.81	7.61
Populating map	2.01	53.24
Merging	58.97	1473.79
Total	**75.79**	**1534.64**

The first step reads the RDF documents, creates data structures on disk, ready for further processing. The second step creates a map on disk, populates it with pairs of UniProt IDs and protein RDF nodes for merging in the third step (together with the matching of genomic sequences).

The total time is displayed in the bottom row. It can be seen that in both versions, about 80% of the total time is spent on the merging phase. This is due to the fact that during merging, for each protein, all relevant RDF triples, including those indirectly related via blank nodes, are extracted from the original RDF graph and then inserted into the merged graph. Effectively, RDF molecules are being identified, created and merged to form a new RDF graph. The diagram below visualizes the time taken by two approaches. Note that JRDF and Sesame take comparable time for preparing the graph and the map. However, the JRDF version of merging takes only about 15% of the time taken by the Sesame version. The savings in integration time are significant and with the increase of the sizes of RDF documents, the difference will become more prominent.

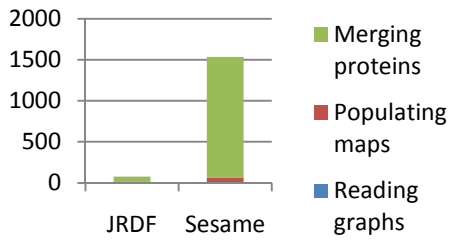

Figure 8 - Comparative Performance of JRDF and Sesame

There are a few differences between Sesame and JRDF's that effect performance. JRDF uses directory structures to keep track of graphs whereas Sesame uses a fourth node and JRDF has a significantly faster localization process.

The graph decomposition algorithm is essentially the same as the protein extraction process. Hence, we can expect with confidence that the decomposition, merging and subsequent querying of the RDF graphs would yield similar high performance.

7. Conclusions

Semantic Web technologies present both enormous promise and significant challenges to the biomedical domain. A number of initiatives and projects (including the W3C Semantic Web Health Care and Life Sciences SIG) have recognized the potential and are embarking on major efforts that involve the representation, integration and reasoning of biomedical datasets using RDF and OWL. However, this work is still at a relatively early stage. There are many problems associated with a lack of standards, tool proliferation, poor maintenance and inadequate and incoherent knowledge representation. This problem is further compounded with inherent limitations in current software and hardware architectures. These have proven to be inadequate for bioinformatics analysis, especially when it requires processing across these rich, open, semantic relationships. Logical reasoning over large or complex ontologies is prohibitively slow – distributed data processing architectures offer a possible solution to this problem.

In this paper, we have presented the BioMANTA project as our proposed solution to the above problem. This paper presents the three major components of the project: (1) a novel "scale-out" architecture – designed to deliver faster, more efficient semantic querying and inferencing of biological data, (2) the BioMANTA ontology, an OWL DL ontology for integrating various protein datasets and (3) the extended molecule and molecule store for the efficient storage, analysis and querying of protein RDF data with blank nodes. The primary advantages that the BioMANTA approach has to offer are:

- The underlying MapReduce architecture distributes the RDF molecules, analysis and semantic inferencing across computational nodes in a cluster to improve scalability and performance, generate cost benefits, and reduce implementation and deployment difficulties. The MapReduce architecture is also easy to maintain and provides a common, powerful and simple way to expedite the storage, processing and analysis of large, complex, heterogeneous biomolecular datasets within a distributed environment.
- The BioMANTA ontology reuses terms from some of the well-established biological ontologies, provides a uniform, semantic representation for protein and protein-protein interaction and pathway data, and supplies vocabularies for the integration of various protein datasets.
- The extensions of hierarchies and ordering to RDF molecules and the RDF molecule store enable the distributed processing of RDF graphs with blank nodes. This approach also helps remove redundancies in RDF graphs.

An initial evaluation of the performance of the prototype implementation has also been presented and shows promising results. However significant further work is required. Future work plans include: a disk-based RDF molecule store to support the storage and retrieval of a larger-scale set of PPI/RDF documents; distribute inferencing using MapReduce processing, distribution of the RDF data over a distributed environment; molecule file serialization format; further evaluation based on exemplary SPARQL queries over the integrated PPI data based on the terms in the BioMANTA ontology; and integration of weightings within the inferencing rules, to reflect the reliability of the source data.

[12] http://biomanta.sourceforge.net/downloads/2008/02/yeast.zip

8. REFERENCES

[1] Ashburner, M., et al., *Gene Ontology: tool for the unification of biology.* Nature Genetics, 2000. **25**: p. 25-29.

[2] Bard, J., S.Y. Rhee, and M. Ashburner, *An Ontology for Cell Types.* Genome Biology, 2005. **6**(2).

[3] Batini, C. and M. Scannapieca, *Data Quality*. 2006, Berlin, Germany: Springer-Verlag. 97--99.

[4] BioPAX Wokrgroup, *BioPAX – Biological Pathways Exchange Language Level 2, Version 1.0 Documentation*. 2005, BioPAX.

[5] Boncz, P.A., *Monet: A Next-generation DBMS Kernel for Query-intensive Applications*. 2002, Universiteit van Amsterdam: Amsterdam, The Netherlands.

[6] Broekstra, J., A. Kampman, and F.v. Harmelen, *Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema*, in *First International Semantic Web Conference*. 2002, Springer: Sardinia, Italy.

[7] Carroll, J.J., *Matching RDF Graphs*. 2001, HP Laboratories Bristol.

[8] Chang, F., et al. *Bigtable: A Distributed Storage System for Structured Data*. in *USENIX'06: Proceedings of the 7th conference on USENIX Symposium on Operating Systems Design and Implementation*. 2006. Seattle, WA: Seattle, WA.

[9] Chatr-aryamontri, A., et al., *MINT: the Molecular INTeraction database.* Nucleic Acids Res, 2007. **35**(Database issue): p. D572-4.

[10] Chu, C.T., et al., *Map-Reduce for Machine Learning on Multicore*. Advances in Neural Information Processing Systems 19, ed. B.S.a.J.P.a.T. Hoffman. 2007, Vancouver, B.C., Canada: MIT Press. 281--288.

[11] Dean, J. and S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. in *SDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*. 2004. San Francisco, CA: USENIX Association.

[12] Ding, L., et al., *Tracking RDF Graph Provenance using RDF Molecules.* Proc. of the 4th International Semantic Web Conference (Poster), 2005.

[13] Floriana Esposito, et al., *REDD: An Algorithm for Redundancy Detection in RDF Models*, in *The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005*. 2005, Springer: Heraklion, Crete, Greece.

[14] Gao, Y., et al., *SWAN: A distributed knowledge infrastructure for Alzheimer disease research.* Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 2006. **4**(3): p. 222-228.

[15] Giovanni Tummarello, C.M., Paolo Puliti, Francesco Piazza. *Signing individual fragments of an RDF graph*. in *Special interest tracks and posters of the 14th international conference on World Wide Web*. 2005. Chiba, Japan: ACM.

[16] Good, B.M. and M.D. Wilkinson, *The Life Sciences Semantic Web is full of creeps!* Briefings in Bioinformatics, 2006. **7**(3): p. 275-286.

[17] Gray, J., et al., *Scientific data management in the coming decade.* ACM SIGMOD Record, 2005. **34**(4): p. 34-41.

[18] Guha, R. *Object co-identification on the Semantic Web*. in *13th World Wide Web Conference*. 2004. New York, USA.

[19] Guldener, U., et al., *MPact: the MIPS protein interaction resource on yeast.* Nucleic Acids Res, 2006. **34**(Database issue): p. D436-41.

[20] Harth, A., et al., *YARS2: A Federated Repository for Searching and Querying Graph Structured Data*. 2007, DERI Galway, Ireland.

[21] Hermjakob, H., et al., *The HUPO PSI's molecular interaction format--a community standard for the representation of protein interaction data.* Nat Biotechnol, 2004. **22**(2): p. 177-83.

[22] Ivanova, M., et al., *MonetDB/SQL Meets SkyServer: the Challenges of a Scientific Database.* Scientific and Statistical Database Management, 2007. SSBDM'07. 19th International Conference on, 2007: p. 13-13.

[23] Jeremy J. Carroll, C.B., Pat Hayes, Patrick Stickler. *Named graphs, provenance and trust.* in *Proceedings of the 14th international conference on World Wide Web*. 2005. Chiba, Japan: ACM.

[24] Joshi-Tope, G., et al., *Reactome: a knowledgebase of biological pathways* Nucleic Acids Research, 2005. **33**(Database Issue): p. 428-432.

[25] Kei-Hoi Cheung, K.Y.Y., Andrew Smith, Remko deKnikker, Andy Masiar, Mark Gerstein, *YeastHub: a semantic web use case for integrating data in the life sciences domain.* Bioinformatics, 2005. **21**(Supp. 1): p. i85-i96.

[26] Kerrien, S., et al., *IntAct--open source resource for molecular interaction data.* Nucleic Acids Res, 2007. **35**(Database issue): p. D561-5.

[27] Khare, R., et al., *Nutch: A flexible and scalable open-source web search engine*. 2004: CommerceNet Labs Technical Report 04.

[28] McNabb, A.W., C.K. Monson, and K.D. Seppi, *MRPSO: MapReduce particle swarm optimization.* Proceedings of the 9th annual conference on Genetic and evolutionary computation, 2007: p. 177-177.

[29] Merelli, E., et al., *Agents in bioinformatics, computational and systems biology.* Briefings in Bioinformatics, 2007. **8**(1): p. 45.

[30] Motik, B., I. Horricks, and U. Sattler. *Bridging the gap between OWL and relational databases*. in *Proc. of the 16th International World Wide Web Conference*. 2007. Alberta, Canada: ACM Press New York, NY, USA.

[31] Muster, P., *Quantitative and Qualitative Evaluation of a SPARQL Front-End for MonetDB*, in *Department of Informatics*. 2007, University of Zurich: Zurich.

[32] Pagel, P.a.K., S. and Oesterheld, M. and Brauner, B. and Dunger-Kaltenbach, I. and Frishman, G. and Montrone, C. and Mark, P. and Stumpflen, V. and Mewes, H.W. and others, *The MIPS mammalian protein-protein interaction database.* Bioinformatics, 2005. **21**(6): p. 832--834.

[33] Pantel, P. *Data Catalysis: Facilitating Large-Scale Natural Language Data Processing*. in *To appear in Proceedings of the International Symposium on Universal Communication (ISUC-07)*. 2007. Kyoto, Japan.

[34] Ronald Read and D. Corneil, *The graph isomorphism disease.* Journal of Graph Theory, 1977. **1**: p. 339-363.

[35] Rubin, D.L., et al., *The National Center for Biomedical Ontology: Advancing Biomedicinethrough Structured Organization of Scientific Knowledge.* OMICS: A Journal of Integrative Biology, 2006. **10**(2): p. 185-98.

[36] Ruttenberg, A., J. Rees, and J. Luciano. *Experience Using OWL DL for the Exchange of Biological Pathway Information*. in *OWL: Experiences and Directions Workshop*. 2005. Galway, Ireland.

[37] Ruttenberg, A., J. Rees, and J. Zucker. *What BioPAX communicates and how to extend OWL to help it*. in *OWL: Experiences and Directions Workshop Series*. 2006. Athens, Georgia, USA.

[38] Ruttenberg, A., J. Rees, and J. Zucker, *What BioPAX communicates and how to extend OWL to help it*, in *OWL: Experiences and Directions Workshop*. 2006: Athens, Georgia, USA.

[39] Salwinski, L., et al., *The Database of Interacting Proteins: 2004 update.* Nucleic Acids Res, 2004. **32**(Database issue): p. D449-51.

[40] Smith, B., et al., *The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration.* Nature Biotechnology, 2007. **25**: p. 1251-1255.

[41] Stickler, P., *CBD - Concise Bounded Description*. 2005.

[42] Stonebraker, M., et al., *C-store: a column-oriented DBMS.* Proceedings of the 31st international conference on Very large data bases, 2005: p. 553-564.

[43] Team, T.Y.R., *Content, Metadata, and Behavorial Information: Directions for Yahoo! Research*, in *IEEE Data Eng. Bull.* 2006. p. 10--19.

[44] Wu, C.H., et al., *The Universal Protein Resource (UniProt): an expanding universe of protein information.* Nucleic Acids Res, 2006. **34**(Database issue): p. D187-91.

[45] Yang, H., et al., *Map-reduce-merge: simplified relational data processing on large clusters.* Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 2007: p. 1029-1040.