BioFederator: A Data Federation System for Bioinformatics on the Web

Ahmed Radwan, Akmal Younis,
Department of Electrical and Computer Engineering
Sawsan Khuri
Miller School of Medicine
University of Miami
a.radwan@umiami.edu,ayounis@miami.edu
skhuri@med.miami.edu

Mauricio A. Hernandez, Howard Ho,
Lucian Popa, Shivkumar Shivaji
IBM Almaden Research Center
mauricio,ho,lucian@almaden.ibm.com
sshivaj@us.ibm.com

Abstract

A problem facing many bioinformatics researchers today is the aggregation and analysis of vast amounts of data produced by large scale projects from various laboratories around the world. Depositing such data into centralized web-based repositories (e.g. NCBI, UCSC Genome Browser) is the common approach. However, the distributed nature of the data, its growth rate, and increased collaborative needs represent real challenges calling for novel decentralized web architectures. The BioFederator is a web services-based data federation architecture for bioinformatics applications. Based on collaborations with bioinformatics researchers, several domain-specific data federation challenges and needs are identified. The BioFederator addresses such challenges and provides an architecture that incorporates a series of utility services. These address issues like automatic workflow composition, domain semantics, and the distributed nature of the data. It also incorporates a series of data-oriented services that facilitate the actual integration of data. The BioFederator is deployed on a grid environment over the web. The proposed design, services, and usage scenarios are discussed in detail. We demonstrate how our architecture can be leveraged for a real-world bioinformatics problem involving tissue specificity of gene expression.

Introduction

This paper presents a life sciences research architecture hosted by the IBM sponsored LA Grid project[1]. LA Grid, an international grid research community, involves collaborations between a number of IBM research centers and many universities from the USA, Latin America, and Spain. It aims to facilitate collaborative research and development amongst participating institutions.

The BioFederator utilizes existing technologies and research results, in an attempt to make progress toward the ever complicated data integration problem in bioinformatics. Specifically, we have componentized Clio (Haas *et al.* 2005), a Java application for schema mapping and data transformation, into several data-oriented services (i.e. schema mapping creation, query generation, query rewriting, query execution, and XML transformation), each wrapped as a web service.

As a proof-of-concept, we utilized Taverna (Hull & et. al. 2006), a bioinformatics workflow tool, to chain together the web services above and solve concrete integration problems in the bioinformatics domain. Using Taverna, one can manually compose a workflow by chaining together a group of web services. However, automatic workflow composition together with capturing domain semantics are two major challenges for a practical bioinformatics data federation system on the web.

To address these challenges we augmented the data-oriented services with a collection of utility services (i.e., coordinator, semantic catalog, repository, and synchronization). The objective is to help automate the process of workflow composition, capture the required domain semantics, and address the distributed nature of the data. The BioFederator is an attempt to define the essential components that would provide an automated, domain-specific, modular, and decentralized data federation system on the web. Nodes of the system can provide data, services (data-oriented or utility services), or a combination of both. Users are able to contribute new data, define new relationships among existing schemas and data sources, relate data to domain-specific concepts, and construct new schemas that others can use.

The main contributions of the study are as follows: First, we present a set of data-oriented web services that are essential for realizing a high-level declarative data federation system on the web. The study details the design, interface, and interaction among these services. Second, we identify a set of essential components and utility services that address the distributed nature of the data, capture domain semantics, and facilitate automatic workflow composition. Finally, in collaboration with bioinformatics researchers within the LAGrid community, we utilize the BioFederator in actual bioinformatics applications and report our observations.

Application

As of September 2006, the Gene Expression Omnibus (GEO) repository at the National Center for Biotechnology Information (NCBI) holds over 3.2 billion measurements deposited by more than 2000 laboratories from around the world (Barrett & et. al. 2006). Public, centralized repositories are the current approach that scientists use to collaborate

[1]Latin American Grid (LAGrid) – http://latinamericangrid.org/index.php

and share their data on a global level.

Centralized repositories like the GEO or the UCSC Genome Browser (Kent *et al.* 2002) for example, are in fact only partially serving these collaborative needs. Bioinformatics is a multidisciplinary field, and as such, scientists from different communities may require a customized view or organization of data. For example, a specific schema definition may be suitable for a computer scientist. However, a biologist may be more comfortable with a different organization of the data. Moreover, the capabilities of relating such views, dynamic sharing, and evolution of data are challenges calling for novel web architectures. The BioFederator is an attempt to address these challenges. It aims to transform public centralized web repositories into a decentralized one that could dynamically evolve, while providing more services. Groups of scientists can dynamically define new schemas, populate these schemas with data, update existing schemas/data, and define relationships between existing schemas/data that others can use.

The BioFederator team involves collaborators from biomedical and genetics domains. The objective of the collaboration is to better understand domain-specific problems and needs, and to ensure that realistic bioinformatics scenarios are addressed. A number of bioinformatics studies illustrate the need for integrating data from multiple sources. The study in (Shaker *et al.* 2004) mentions some examples. Pharmacogenomics is yet another example of an emerging arena with increasing computational needs. Pharmacogenomics deals with the influence of inherent genetic variation on drug responses in patients, promising the advent of "personalized medicine" in which drugs and drug combinations are optimized for each individual's unique genetic makeup. To make such "personalized" medical decisions, information from multiple heterogeneous data sources needs to be retrieved and analyzed in a quick, user-friendly fashion. The databases that are currently available that house pieces of this information puzzle are diverse: for example, OMIM (McKusick 1998) and dbSNP (Smigielski *et al.* 2000) from NCBI, the TRANSFAC database from BioBase (Kel-Margoulis & et. al. 2005), and PharmGKB (Klein & et. al. 2001).

Figure 1 shows a hypothetical study that aims to understand the rate of gene expression in different tissues, and to correlate these expression profiles with active transcription factors and their binding sites. The data required for this study is distributed among multiple heterogenous sources (e.g., UCSC Genome Browser, GNF SymAtlas (Su & et. al. 2002), and TRANSFAC). The figure shows how Clio mapping technology can be used to provide a high-level definition for mappings between the source and target schemas. In particular, a graphical user interface allows entering correspondences that relate schema elements. The *schema mapping creation service* takes the correspondences as input and computes a more precise mapping specification based on the schema information and constraints. This mapping specification can subsequently be used by other services. More details on the specific data services are provided in the *Architecture and Services* section, and this figure will be revisited in more detail in the *Usage Scenarios* section.

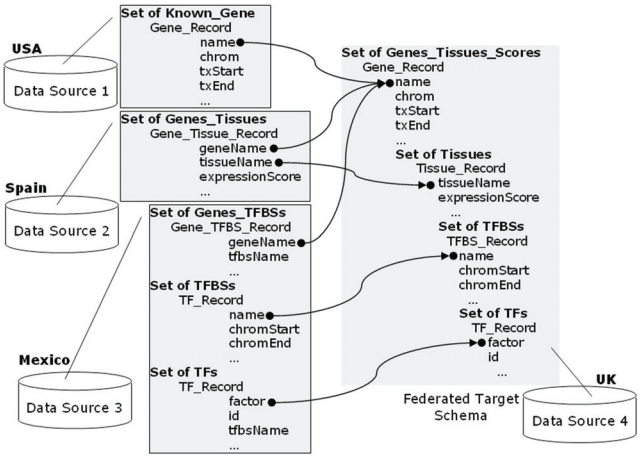

Figure 1: High-level mapping definition using Clio

It should be noted that the BioFederator focuses on bioinformatics data integration applications. However, a typical bioinformatics research project involves both computational and data driven aspects. While data driven processes may extract data from various sources, the computational processes would process the data through algorithms for pattern matching, sequence alignment, and clustering. A computational architecture and associated services for the bioinformatics domain is under investigation.

Related Work

A number of data integration systems have been proposed to address the problem of large-scale data sharing, e.g., (Garcia-Molina & et. al. 1997), (Duschka & Genesereth 1997), (Manolescu, Florescu, & Kossmann 2001), and the survey by Halevy (Halevy 2001)). These systems support rich queries over large numbers of autonomous, heterogeneous data sources by making use of semantic relationships between the different source schemas and a *mediated schema*, that is designed globally. However, the mediated schema itself becomes a problem. First, it may be hard to come up with a single mediated schema which everyone agrees with. Second, all the access (querying) is done via a single point (the mediated schema). Furthermore, this architecture is not robust with respect to changes in the source schemas. As a result, data integration systems based on mediated schemas are limited in supporting large-scale distributed and autonomous data sharing.

Peer Data Management Systems (PDMS), e.g., Piazza in (Halevy *et al.* 2004), have been proposed to address the aforementioned problems and to offer an extensible and decentralized data sharing system. The BioFederator's requirements are, in principle, no different from these peer data management systems. Compared to Piazza, our intended applications imply smaller numbers of data sources. However, the sources have complex schemas and may contain overlapping and potentially conflicting and dynamically changing data. BioFederator emphasizes the use of tools and services that facilitate mappings among schemas and generate

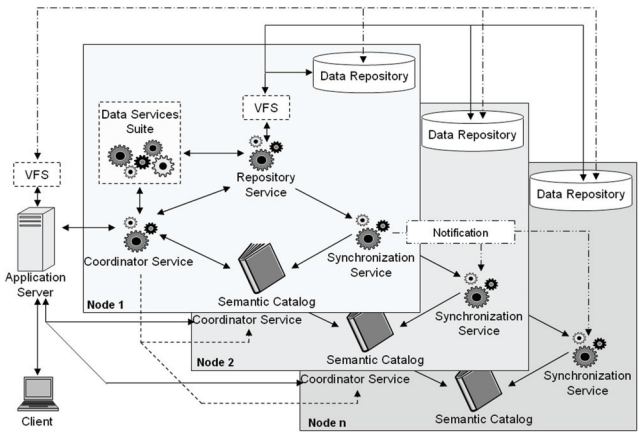

Figure 2: BioFederator nodes architecture

the queries that are needed to access and integrate the data. In many respects our vision is similar in spirit to that of the SHARQ project (Taylor & Ives 2006).

Architecture and Services

The system is built from a collection of nodes, a node can be any device or computer having the required web services deployed. Nodes can be dynamically added or removed, and the system adapts to deliver quality of service requirements.

Figure 2 depicts the architecture of typical BioFederator nodes. Each node contains six components/services, namely: the *coordinator service*, the *semantic catalog*, the *repository service*, a local *data repository*, the *synchronization service*, and the *data services suite*. To support system portability, all components and services are built using Java and XML. The proposed system architecture supports distributed storage and manipulation of data. One or more application servers can connect users to the BioFederator. The application server can initiate multiple concurrent requests (by contacting multiple nodes). However, for every request, the application server designates a particular node as the "master" node. Any node is capable of performing the "master" role. The master can distribute the required operations among many other nodes and is also responsible for coordinating, collecting, and merging results from "slave" nodes.

Coordinator Service: This service is the front-end interaction point for a given BioFederator node. It maintains a list of all registered nodes in the whole system, in addition to locally and remotely deployed services. It can access the metadata maintained in the local semantic catalog (discussed later in detail). Coordinator services are key elements for automated workflow construction because they are responsible for decisions involving forwarding, splitting, or directly handling a received request.

Upon receiving a request, the coordinator needs to access the semantic catalog to retrieve information about data needed/involved in the request. Based on metadata and information about registered nodes and deployed services, the coordinator can make a decision on handling the request, i.e., forward, split, or directly handle.

The coordinator maintains a set of rules to help make such decisions. The efficiency and accuracy in defining such rules is crucial for correct and efficient system performance. The rules are a function of the quality of service, load balancing, and optimization requirements. While a system could function well using limited rules, its performance could be enhanced by adding and tuning rules. For example, a simple rule is to forward the request to the first node that has the required services deployed. However, a better rule is to incorporate the location of the data. From our experience, fine tuning rules may result in complex but efficient workflows of services. Policies like query distribution and data materialization are initiated by the *coordinator service*.

Semantic Catalog: This service provides a domain-specific organization of the data. Figure 3 depicts the structure of an illustrative subset of the semantic catalog. The BioFederator does not require a fixed structure for the semantic catalog. In fact, different structures could arise based on application needs and it can dynamically evolve over time. The structure used and illustrated in Figure 3 is inspired by the one utilized at the UCSC Genome Table Browser[2].

The semantic catalog in Figure 3 is organized as a set of topics in a tree structure; the root node is the most general topic, i.e., the whole catalog, while leaf nodes represent the most specific topics, i.e., schema definitions. Topics are labeled and each has an associated unique identifier, `TID`. This can be evaluated by traversing the tree starting from the root, the root topic has `TID=0`. The depicted dark route in the figure highlights the path traversed to evaluate the `TID` for the *"Expression/Regulation"* topic (`TID=0.2.2.1.2`).

The tree structure facilitates an XML representation of the catalog with the associated query mechanisms, and a synchronization mechanism based on WSRF[3] notification. However, other structures could be used if the appropriate query and synchronization mechanisms are provided.

Topics play an important role in the devised notification mechanism (discussed later in the *synchronization service*). Every node in the proposed architecture defines its own set of "topics of interest". Individual topics are only replicated on nodes that identify them as "topics of interest". That is, the system does not replicate the entire catalog on all nodes. This technique has three advantages: 1) It limits the size of the semantic catalog on specific nodes, 2) Topics can help large systems define specialized clusters of nodes, and 3) It helps preserve autonomy of nodes (each node has full control of the contents of it's local catalog). Note that, if all nodes identify the entire catalog (`TID=0`) as a topic of interest, then the full catalog will be replicated on all nodes.

Figure 3 illustrates the distributed data repository and depicts how semantic catalog topics are linked to data resources stored there. Leaf topics represent schema definitions, and they can point (using URIs[4]) to one or more data

[2] http://genome.ucsc.edu/cgi-bin/hgTables

[3] http://www.globus.org/wsrf/

[4] Uniform Resource Identifiers

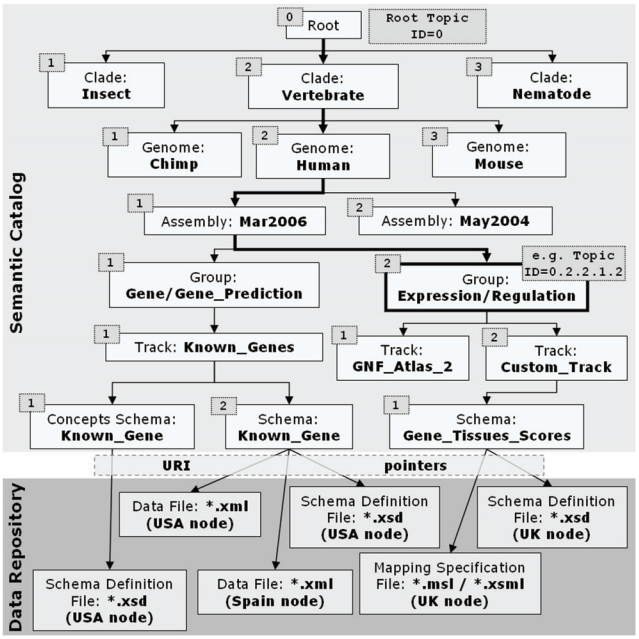

Figure 3: The semantic catalog structure and relation with data repository elements

resources. A basic URI is needed to point to the schema definition (XSD) file, and additional URIs can point to one or more instances of this definition, i.e., XML files. In addition, pointers to schema mapping files that directly relate pairs of schemas can be specified. Such mappings, created by the *schema mapping creation service* described later, specify the exact semantics of data conversion from one schema to another, and can be subsequently used by other data services.

Two kinds of schema topics are identified in the semantic catalog illustrated in Figure 3, specifically the *schema* and the *concepts schema*. A *schema* follows the conventional definition while a *concept schema* is created by mapping schema elements to concept unique identifiers (CUIs) from a specific conceptual model, e.g., using the UMLSKS[5] Metathesaurus. For example, a schema element representing a *gene identifier* can be named *geneID* in one schema and *gID* in another, however both will be mapped to the same UMLSKS CUI: *C1706509*. Typically, the conventional schema and its corresponding concepts schema exhibit the same structure. Mutual correspondences between conventional and concept schemas are stored in the semantic catalog. Concept schemas can be useful for deriving relations between different schema definitions, e.g., to discover if a given schema is equivalent, a superset, or a subset of an existing one.

Repository Service and the Data Repository: Currently, this service is implemented on top of the Apache Commons Virtual File System (VFS)[6]. VFS provides APIs for accessing different file systems and presents a unified view of files from different sources (e.g., local disk, remote ftp, or http servers). The *repository service* is responsible for storing and extracting all raw data (via the VFS). It also notifies the *synchronization service* if there are any changes to the local repository. In the current implementation, only a pure XML data repository is supported. Other data representations can be supported if the data can be exported as XML.

Synchronization Service: This service keeps the local semantic catalog entries synchronized with other nodes. When a node is added, it has the option of subscribing to various topics. Whenever a change occurs in the semantic catalog (of one node) affecting a certain topic, nodes that are subscribing to that topic receive notifications of the update. This service is implemented on top of the WSRF notification mechanism provided by the Globus Toolkit.

Data Services Suite: This component provides a number of web services that allow the creation of schema mappings and operations using those schema mappings. We have selected Clio's (Haas *et al.* 2005) schema mapping components, and wrapped them into web services. The suite provides the following data services:

- **Schema Mapping Creation**: Given a source and a target schema, and a set of "correspondences" between source and target schema elements, this service creates a "mapping" from the source to the target schema. This mapping consists of a set of declarative constraints that dictate what the target instance should be, given a source instance. The mapping creation algorithm takes into account the schema constraints, e.g., foreign key constraints and type constraints, as well as the correspondences (Popa *et al.* 2002).
- **Query Generation**: Given a mapping produced by the Schema Mapping Creation service, this service generates an XQuery, XSLT, or a SQL/XML query that implements the transformation implied by the mapping (Haas *et al.* 2005). The query and its associated mapping are stored in the semantic catalog, potentially for further reuse.
- **Query Execution**: For convenience, we also have a service that executes the queries generated by the previous service. Given a query script and a set of input XML documents (instances of the source XML schema used in the mapping, for example), the service executes the query and returns the resulting XML document.
- **XML Transformation**: We also provide a service that allows the direct and scalable execution of the mapping, as opposed to simply executing the query that implements it. Based on the technology detailed in (Jiang *et al.* 2007), this service takes as input a mapping and the source XML instances and returns the target XML instance that is implied by the mapping. As opposed to the Query Generation or Execution services, this service neither produces nor executes a query; rather, this service uses a Java based engine that optimally executes the mapping.
- **Query Rewrite**: An interesting application of mappings is the ability to rewrite (or translate) target-side queries

[5]Unified Medical Language System Knowledge Source Server, http://umlsks.nlm.nih.gov

[6]http://jakarta.apache.org/commons/vfs/

into queries that work on the source-side. This is useful, for example, if the target side schemas are virtual and the actual data resides on the source side. We use the query rewriting techniques detailed in (Yu & Popa 2004) to implement this service. Given a schema mapping and an XQuery over a target schema instance, this service returns a rewritten XQuery over the source schemas in the mapping. When the rewritten XQuery is executed against the data source instance, the resulting data has the same structure as the intended output of the original target XQuery.

- **Schema Integration**: Given a number of mappings between several schemas, this service attempts to create an "integrated" schema that captures the unified concepts of the schemas that are related by the mapping[7] (Chiticariu, Kolaitis, & Popa 2007). For example, given a mapping between two schemas S1 and S2, this service creates a new schema T that integrates the concepts in S1 and S2. This can eliminate the need to manually construct such target schema.

Usage Scenarios

We present an example of a collaborative scenario that can be achieved using the BioFederator. To simulate this scenario, schemas and data were downloaded from public repositories, and were then stored on various BioFederator nodes. The hypothetical scenario involves three collaborating groups of scientists. Assume that the groups are associated with the three data sources shown in Figure 1, and are located in the USA, Spain, and Mexico respectively. The group based in the USA is conducting experiments related to the identification of known genes, their chromosomal positions, and so on. The team in Spain is investigating gene expression levels in different tissues, while Mexican team is concerned with the identification of transcription factor binding sites that are active on different genes, and the associated transcription factors. Furthermore assume that there is a fourth team in the UK that will do the analysis of the collected data. Their role is to collect and interpret data from different teams and to discover new knowledge from the experiments.

Initialization: Each site in the study independently sets up and configures its own node. The configuration process includes the installation of any required software and the deployment of required services. As nodes join the system, their coordinator services update lists of registered nodes, deployed services (both local and remote), and subscribe to the "topics of interest".

Data Storage: Assume that the group in the USA has collected data that needs to be deposited into the system. The first step is to define a suitable schema to represent this data, if one is not already available. The next step is to prepare the actual data instance conforming to the designed schema and choose the associated parent topic (the topic should be identified in the hierarchy of topics in the *semantic catalog*). Optionally, mapping from schema elements to CUI, concept unique identifiers, can be provided to help construct the corresponding *concepts schema*.

Having the schemas and actual data, the USA team connects to their node via an application server and starts uploading their data. The *coordinator service* delegates the *repository service* which handles the storage process (via VFS) and notifies the *synchronization service* to update the local *semantic catalog*. Figure 3 shows the uploaded schema *Known_Gene* topic pointing to the schema definition file and actual instances on the USA node. The *synchronization service* also notifies remote synchronization services, which are subscribed to the topic associated with the newly uploaded data. Note that the *coordinator service* may decide to store data on multiple nodes based on quality of service requirements. Figure 3 shows an additional instance saved on the Spain node.

Schema Mapping: Assume now that the teams in Spain and Mexico have also uploaded their data. Thus, all three data sources have been loaded with data, which is hosted on different nodes. Now the analysis team in the UK can start interpreting and analyzing the data deposited by the other three groups. However, they are facing the problem of how to merge and integrate this data. They realize that in order to efficiently analyze the data, they would like to organize it according to a specific structure. Therefore, the UK team constructs a new schema that captures the required data organization (the target schema in Figure 1).

The BioFederator includes a powerful schema mapping component that can create the mappings from the source schemas into the new target schema. The UK team connects and downloads (via Java Web Start) an application that allows the construction of Clio-based mappings. Source and target schemas are loaded into the tool which shows their structure as a tree of schema elements (very similar to how they are presented in Figure 1). Value mappings are entered by drawing lines from source schema elements to target schema elements. The *schema mapping creation service* processes the mapping specification and passes it to the *repository service* for storage. The *synchronization service* updates the local *semantic catalog* and notifies remote nodes about the new mapping. Figure 3 shows the target schema "*Gene_Tissues_scores*" pointing to both the schema definition file and the mappings file on the repository (UK node). When a decision is taken to construct a materialized instance of the target schema, the mappings specifications file is read and the source schemas TIDs are extracted.

Query Processing: After constructing the new, federated target schema in the previous step, various analysis groups can start accessing and querying it.

Different query processing scenarios could arise based on the location of data (which can be either locally or remotely stored), and whether the query is against a materialized version of the data or not. If the data is not materialized then either a materialization or a query distribution decision could be made by the *coordinator service*. Criteria for this decision can be based on the frequency of the queries against the data sources, and it can be locally materialized if the number of queries received exceeds a specific threshold.

For instance, imagine that the UK team is trying to answer

[7]The service currently provides a ranked list of possible integrated schemas for the user to choose from.

the following query using the federated target schema:

> Find a list of **genes names** and **their chromosomal locations** having an **expression level** $\geq e$ in both **heart** and **liver**, and that are **regulated by the same set of transcription factors**.

The above query is written against the federated target schema. However, the UK node does not have any data associated with this federated schema, i.e., all data resides at the other nodes (a *Global-Local-As-View* (GLAV) scenario (Friedman, Levy, & Millstein. 1999)). One alternative could be data materialization, i.e., creating an instance of the target schema, using *query generation* and *execution services*, and then executing the query against this instance. Another alternative is using the *query rewrite service* as discussed above.

Conclusion and Future Work

This paper introduces the BioFederator, a web services-based data federation system. The data-oriented services offer high-level data mapping and transformation capabilities while the utility services address the distributed nature of the data, capture domain semantics, and automate the workflow composition process. The presented architecture represents a novel, modular, and decentralized method to realize web based bioinformatics repositories. We are planning to extend the current pilot implementation by involving more bioinformatics research groups and scenarios to identify the capabilities, limitations, and potential future developments.

References

Barrett, T., and et. al. 2006. NCBI GEO: mining tens of millions of expression profiles database and tools update. *Nucleic Acids Research* 35:D760–D765.

Chiticariu, L.; Kolaitis, P. G.; and Popa, L. 2007. Semi-Automatic Generation and Exploration of Schema Integration Alternatives. Submitted for publication.

Duschka, O. M., and Genesereth, M. R. 1997. Answering recursive queries using views. In *PODS*, 109–116.

Friedman, M.; Levy, A.; and Millstein., T. 1999. Navigational plans for data integration. In *16th National Conference on Artificial Intelligence (AAAI)*.

Garcia-Molina, H., and et. al. 1997. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems* 8(2):117–132.

Haas, L. M.; Hernández, M. A.; Ho, H.; Popa, L.; and Roth, M. 2005. Clio Grows Up: From Research Prototype to Industrial Tool. In *SIGMOD*, 805–810.

Halevy, A. Y.; Ives, Z. G.; Madhavan, J.; Mork, P.; Suciu, D.; and Tatarinov, I. 2004. The Piazza Peer Data Management System. *IEEE Trans. Knowl. Data Eng.* 16(7):787–798.

Halevy, A. 2001. Answering Queries Using Views: A Survey. *VLDB Journal* 270–294.

Hull, D., and et. al. 2006. Taverna, A tool for building and running workflows of services. *Nucleic Acids Research* 34:w729–w732.

Jiang, H.; Ho, H.; Popa, L.; and Han, W.-S. 2007. Mapping-Driven XML Transformation. In *16th International World Wide Web Conference*.

Kel-Margoulis, O., and et. al. 2005. *Information Processing and Living Systems:Databases on Gene Regulation*. Imperial College Press, London.

Kent, W.; Sugnet, C.; Furey, T.; Roskin, K.; Pringle, T.; Zahler, A.; and Haussler, D. 2002. The Human Genome Browser at UCSC. *Genome Research* 12(6):996–1006.

Klein, T., and et. al. 2001. Integrating Genotype and Phenotype Information: An Overview of the PharmGKB Project. *The Pharmacogenomics Journal* 1:167–170.

Manolescu, I.; Florescu, D.; and Kossmann, D. 2001. Answering XML Queries on Heterogeneous Data Sources. In *VLDB*, 241–250.

McKusick, V. 1998. *Mendelian Inheritance in Man. A Catalog of Human Genes and Genetic Disorders*. Baltimore: Johns Hopkins University Press.

Popa, L.; Velegrakis, Y.; Miller, R. J.; Hernández, M. A.; and Fagin, R. 2002. Translating Web Data. In *VLDB*, 598–609.

Shaker, R.; Mork, P.; Brockenbrough, J. S.; Donelson, L.; and Tarczy-Hornoch, P. 2004. The BioMediator System as a Tool for Integrating Biologic Databases on the Web. In *Proceedings of the Workshop on Information Integration on the Web*.

Smigielski, E. M.; Sirotkin, K.; Ward, M.; and Sherry, S. T. 2000. dbSNP: a database of single nucleotide polymorphisms. *Nucleic Acids Research* 28(1):352–355.

Su, A., and et. al. 2002. Large-scale analysis of the human and mouse transcriptomes. In *National Academy of Sciences of the United States of America*.

Taylor, N., and Ives, Z. 2006. Reconciling while tolerating disagreement in collaborative data sharing. In *SIGMOD*, 13–24.

Yu, C., and Popa, L. 2004. Constraint-Based XML Query Rewriting for Data Integration. In *SIGMOD*, 371–382.