

# DataFoundry: Information Management for Scientific Data

T. Critchlow, K. Fidelis, M. Ganesh, R. Musick, T. Slezak

**Abstract** - Data warehouses and data marts have been successfully applied to a multitude of commercial business applications. They have proven to be invaluable tools by integrating information from distributed, heterogeneous sources and summarizing this data for use throughout the enterprise. Although the need for information dissemination is as vital in science as in business, working warehouses in this community are scarce because traditional warehousing techniques don't transfer to scientific environments. There are two primary reasons for this difficulty. First, schema integration is more difficult for scientific databases than for business sources, because of the complexity of the concepts and the associated relationships. While this difference has not yet been fully explored, it is an important consideration when determining how to integrate autonomous sources. Second, scientific data sources have highly dynamic data representations (schemata). When a data source participating in a warehouse changes its schema, both the mediator transferring data to the warehouse and the warehouse itself need to be updated to reflect these modifications. The cost of repeatedly performing these updates in a traditional warehouse, as is required in a dynamic environment, is prohibitive. This paper discusses these issues within the context of the DataFoundry project, an ongoing research effort at LLNL. DataFoundry utilizes a unique integration strategy to identify corresponding instances while maintaining differences between data from different sources, and a novel architecture and an extensive meta-data infrastructure, which reduce the cost of maintaining a warehouse.

## 1. Introduction

Over the past several years, there has been an explosion in the amount of information available to both business and scientific enterprises. Individuals and corporations routinely use computers both to record proprietary information and to distribute public information via the WWW. The challenge facing data managers today is how to fully utilize this wealth of information without overwhelming either the end user or the system maintainers. Because of the competitive advantage provided by better data analysis, business information processing has driven technological advances in data warehousing and analytic processing. Within this community, relational databases are accepted as standards for transaction-based systems and SQL provides a consistent, well-known data access and manipulation language. Several commercial tools, from companies such as Red Brick, Cerebellum, Sagent, Informatica, VIT, and many others, address the steps

involved in creating, maintaining, and analyzing a warehouse.

Unfortunately, scientific applications face unique problems that are not being addressed by those tools. While part of the problem arises from the lack of standardization in scientific domains – for example, information sources do not share a common terminology, data representation, or data management architecture – the primary problems are the subtle but complex relationships between data and the dynamic source schemata. DataFoundry is an ongoing research effort at LLNL focused on making data warehousing feasible for scientific environments. DataFoundry has concentrated on bioinformatics both because these problems are rampant in that domain, and because failure to unify the publicly available data sources will hamper scientific progress in that field; however, the solutions that we propose are appropriate for most scientific domains.

One of the long-term goals of bioinformatics is to enable advanced computer analysis for identifying similarities (homologies) between DNA and protein sequences and structures. A specific and significant application of this analysis would be in the area of protein fold recognition and comparative modeling. Research in this area is based on the hope that a better understanding of sequence patterns, protein structures, and the complex relationships between them will make structure prediction feasible. Because full-scale physics modeling is currently impractical, this knowledge must be obtained by careful examination of known protein structures, folding properties and propensities, protein and DNA sequences, and genetic organization and expression data. While much of this information is available on the Web, it is spread across multiple community databases, each using its own concepts, semantics, data formats, and access methods. Currently, the burden falls on the scientist to resolve conflicts, integrate the data, and interpret the results. More often than not, this barrier proves too difficult to overcome, and data is under-exploited.

The goal of DataFoundry is to reduce the cost of creating and maintaining a semantically consistent view of the data from several dynamic, heterogeneous data sources. To evaluate our approach, we have developed a prototype data warehouse currently in use by structural biologists at LLNL. The prototype has three levels of interface: web-based forms for the novice user; a graphical query editor for intermediate level users; and direct SQL access through C++ and Perl for the more advanced user. To date, Protein Data Bank (PDB) [5], SWISS-PROT [1], SCoP [11], and dbEST [3] data have been integrated into this framework. Additional sources will be incorporated, allowing us to support not only

protein structure research, but also functional genomics. DataFoundry is distinguishable from other efforts in the bioinformatics community, and the commercial data warehousing world, by three salient points:

- 1) *A unique association strategy* – By linking related inter-database data, as opposed to integrating them by defining a single, correct view, we provide consistent views of instance data when appropriate while still maintaining the heterogeneity between the instances.
- 2) *A novel architecture* – We have designed an architecture that improves both reliability and performance (when compared to a federated architecture), while still providing full access to the original data.
- 3) *A meta-data based infrastructure* – We have defined a meta-data based mediator generator that substantially reduces the cost of creating and maintaining a warehouse.

This paper describes the challenges involved in successfully performing schema and data integration in highly dynamic environments, and presents DataFoundry's approach to addressing them. While there are several other important issues that must be resolved in any information management venture, such as data cleaning and consistency, we do not have sufficient room to explore them here. We begin with an overview of related efforts in the next section. Section 3 briefly discusses the problems associated with current schema integration techniques before presenting our approach. Section 4 describes the implications of architecture choices on warehouse maintainability, and finally, Section 5 briefly discusses our meta-data infrastructure.

## 2. Background

Scientific applications face problems not being addressed by commercial tools. In domains such as genomics, information sources do not share a widely accepted format for storage or access; some rely exclusively on flat files while others utilize relational or object-oriented databases. Moreover, the types of information provided by these sources, and the corresponding data representations, are continuously evolving. Few existing tools allow the modeling and management of the complex data encountered in scientific applications. In this section, we look at some of the approaches to data management in bioinformatics and compare them with DataFoundry.

The Object Protocol Model (OPM) [6], developed at Lawrence Berkeley National Laboratory, provides a set of tools for scientific data management. Representing scientific experiments requires more complex data models and modeling facilities than are provided by commercial relational database management systems (RDBMSs). Thus OPM uses an object-oriented data model and includes a *protocol* class for modeling experiments. The OPM toolset consists of a schema editor for specifying and managing OPM schemata, a graphical querying and browsing tool, and tools for conversion between relational and OPM schema definitions to work with existing scientific databases.

The OPM toolset has been extended with facilities for querying a multi-database system linking several databases. This toolset provides complimentary functionality to DataFoundry, in that it addresses a different set of bio-informatics challenges.

The CPL/Kleisli [13] project (U Penn) also provides tools to manage the transformation of data between databases, and to provide integrated access to multiple data sources. The transformation and integrated access is achieved through constraints and rules specified in special purpose languages. Kleisli follows a multi-database approach to integrated access across multiple data sources. The multi-database approach does not provide an integrated schema across the source databases. Hence users are required to directly specify the rules and constraints involved in queries for integrated access to the databases. While extremely general, this approach prevents casual users, who are unfamiliar with the details of the individual data sources, from fully utilizing the resource.

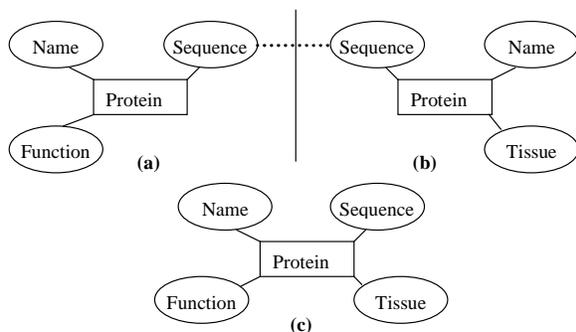
The aim of The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS) [10] is to provide tools for integrated access to multiple information sources. TSIMMIS uses a self-describing object model, called the Object Exchange Model (OEM), and wrappers to translate between OEM and native database languages/models. Mediators are used as query managers to locate the data sources containing the requested information. The thrust of this project is to provide access to diverse and dynamic information sources which are often unstructured or semi-structured such as the World Wide Web. The usefulness of this approach in a scientific application will depend greatly on how easily data in the particular domain can be represented using OEM.

The EasyQuery program [15] from CyberConnect provides graphical query access to multiple databases in the biological domain. The query editor has facilities to import and display the relational schemas of geographically distant community databases. Users can then pose queries on these schemata, which will be executed at the corresponding sites. Meta-data is maintained about the participating databases. This is used to provide information to users on correct ways of combining information across databases using join queries. This approach requires the use of hard coded information on relationships between participating databases: adding new databases requires updating inter-database relationships.

There are many other approaches to interacting with scientific data – for example, web based forms – however, these approaches significantly limit the user's ability to interact with the data and are not considered further.

## 3. Integrating Heterogeneous Data Source Schemata

DataFoundry uses a global schema to present a coherent view of the data from several sources. Defining such a schema involves identifying related concepts in different data sources, then mapping them to a common format, while resolving the syntactic and semantic



**Figure 1 Simple Schema Integration**

differences that occur between the individual representations. Traditionally, a global schema is created by conceptually overlaying the source schemata, with similar concepts and attributes superimposed. Once the global schema has been identified, translation functions, or mediators, are created to map source concepts to the global representation. While this approach provides a consistent view of the data to the users, it defines an implicit relationship between attributes from different data sources. This relationship may produce unintended results either because of difficulty identifying corresponding instances between the databases (the object identity problem) [9][12] or difficulty understanding how to merge data assumed to be identical but that actually contains minor differences [1].

For example, consider the protein representations shown in Figure 1(a) and (b). They both include name and sequence information. However, the first also contains the protein's function, while the second includes the tissue that the protein was found in. If these concepts are overlaid, the resulting composite representation is shown in Figure 1(c). While this is a reasonable representation of the concept, serious problems may arise in practice because of the implicit relationship between the attributes from different data sources. Consider a protein in database (a) that is carcinogenic, and its corresponding entry in (b) which states that it occurs in the breast. While these instances may share same name, they could have slightly different sequences. This type of mismatch is common in both business and scientific domains. The important distinction is that while in business there is a single, correct value, this is not always the case in scientific domains.

When these databases are combined, the mediator must decide whether these instances represent the same protein. If the decision is that they do, they will be combined in the resulting global view; otherwise they will remain distinct. Neither choice is entirely acceptable. If the instances are combined, the mediator must select one of the original sequences to represent the resulting instance. As a result, some of the heterogeneity information is lost. Conversely, if the instances are not combined, the association between the function of the protein and the corresponding tissue is not identified. Thus, queries such as *return the sequence of all cancer related proteins found in the breast* will not produce the expected results. This simple example suggests

resolving the conflict by adding a second sequence attribute to the global definition, and mapping each of the original sequences to exactly one of these global attributes. While this initially appears to be a reasonable solution, the complexity of the resulting queries and data structures quickly becomes too complex to manage in the face of multiple data sources. To reduce the potential for misunderstanding, we have taken a different approach which is described in the remainder of this section.

### 3.1. Natural Keys

Instead of overlaying complex concepts, such as proteins, *primitive* concepts likely to be represented within multiple data sources are identified. Inter-database relationships are permitted only between these concepts. These base concepts form *natural keys* between databases, and represent fundamental real world concepts that are intrinsically sharable and have precisely defined semantics. Identifying natural keys requires careful examination of each data source as well as an expert understanding of the underlying concepts. If overly simplistic concepts are chosen (e.g. residues), it is extremely difficult to identify and represent interesting inter-database relationships. On the other hand, if complex concepts are chosen, implicit relationships may be created resulting in the problem previously described. We have found that natural keys are usually a single characteristic associated with all definitions of a more complex concept. While each definition may use different semantics and data formats to represent these characteristics, the semantics will be closely related and easily generalized to a single, inclusive definition. For example, when integrating PDB and SWISS-PROT, we identified a protein sequence as a natural key: protein sequences are a basic concept found in several data sources, with obvious mappings between the various representations.

### 3.2. Inter-Database Correspondences

We have identified three categories of inter-database relations<sup>1</sup> based on natural keys: traditional relationships, identity, and similarity. Traditional relationships between natural keys, for example *person A wrote article B*, are represented as normal relationships within the global schema, and are instantiated on the local data store. Since we treat these relationships in the traditional way, they are not discussed further.

The identity relationship relates those instances of natural keys in different databases that *appear* to represent the same object. This relationship explicitly defines the association between natural keys which is implicitly defined in the overlay approach. By making the relationship between corresponding instances explicit, we are able to preserve data heterogeneity while maintaining the connection between all attributes associated with a given concept. Consider again the two

<sup>1</sup> Relationships that occur solely within a single database are trivially included within the global schema when the enclosing source schema is integrated

structures represented in Figure 1(a) and (b); in our approach the global schema would contain both original structures related through this relationship on the sequence attribute (shown by the dotted line). The heterogeneity present in the sources remains, but the correspondence between the tissue and the function is also identified. Sometimes it is useful to have a less stringent association between natural keys; for example, sequences that are homologous but not identical. For these cases, the similarity relation is used. Similarity identifies an association between attributes that are probably not the same, yet are weakly related through some statistical equality operation. As a more concrete example, consider the protein sequences that are used as natural keys for PDB and SWISS-PROT. Sequence homology is used to identify both identical proteins, based on exceptionally high homology scores, and similar proteins, based on significant but lower scores. Because we allow minor variation in sequences to be counted as identical proteins, we can provide users with all of the information related to a specific protein while still maintaining the heterogeneity found in the sources. The similarity relation allows us to identify closely related proteins, such as those in the same family.

Because natural keys are fundamental concepts, their representation is unlikely to change significantly, even if the source schema is modified. By limiting inter-database correspondences to these concepts, the effects of a schema change can be isolated to the small portion of the global schema corresponding to the data source. At the same time, the association between identical concepts provides a rich environment within which experienced users can ask complex queries over all the data. Creating explicit relationships between these concepts significantly reduces the effort required to adapt to schema modification, while permitting all of the information about an instance to be easily identified and retrieved.

#### 4. The DataFoundry Architecture

Traditionally, three approaches have been used to access data from multiple external, heterogeneous data sources: multi-databases, federated databases, and data warehouses. Multi-databases [1] provide a simple connection between systems, permitting the user to create queries across multiple databases at the same time. Unfortunately, because this approach does not provide a consistent view of the data, users are expected to formulate extremely complex queries. In particular, for each query, a user must understand the internal representation of each relevant source, manually resolve syntactic and semantic conflicts, and construct queries using the sources' native query language. Unless every potential user is intimately familiar with the detailed workings of each connected data source, this is not a desirable approach.

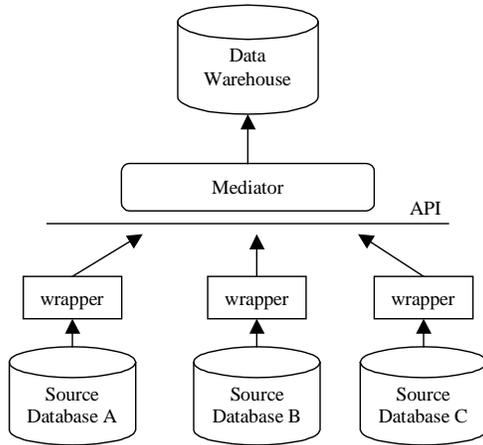
When providing a resource to a broad community, it is critical to provide a consistent interface to the data. To this end, federated databases [14] define an integrated schema over the subset of available data that is interesting to the federation as a whole. This global schema represents a virtual database, combining

data from each participating source to form a single, consistent representation. Queries posed over a federated database are sent to the applicable data sources, after being translated into the native query language, and the results are combined before being passed on to the user. There are three drawbacks to this approach. First, in order to process queries in an efficient way, sources participating in a federation may be required to contribute resources (e.g. query capabilities, storage facilities) to the federation. Second, because data is not represented locally, this approach is susceptible to long delays when answering queries. Third, misleading or incorrect results may be returned when a data source is unavailable, even temporarily.

To reduce the amount of network traffic and improve query results, data can be combined in a single database – resulting in the traditional, monolithic data warehouse. This approach introduces two new problems: first is the tremendous amount of storage required to keep all of the data (in some cases several terabytes) within a single database. To avoid this problem, warehouses typically contain only summary and aggregate data. Obviously, warehouses using this summarized data provide very different types of information than the original data sources, and may be of limited use in a scientific environment. Second is the difficulty keeping the data current and the warehouse fully functional. In business domains, such as accounting, the source schemata are quite stable (e.g. the notion of an account rarely changes). In dynamic scientific areas such as genetics, existing entries may change based on new information, and source schemata are rapidly evolving with our understanding of the underlying biology.

DataFoundry combines many of the advantages of the systems described above to form a unique approach to database integration. It provides a consistent view of integrated data to the users. It uses a local data cache to reduce network traffic and improve performance and reliability. DataFoundry is based on a mediated data warehouse architecture [10], shown in Figure 2, which is a combination of the monolithic warehouse and federated database approaches. This architecture provides a global schema containing a subset of the information in the data sources, similar to a federated database. However, the DataFoundry schema is expanded to include some summary and aggregation information. A local data store contains both this additional data, and the most important and most frequently accessed source data. The result is a consistent view of the data and greatly improved query performance, with the ability to pose ad hoc queries against the original domain data.

The subset of data cached locally is initially identified when the source is first integrated, but may be refined over time. Queries involving data not represented in the local store require accessing a remote site, as in a federated database. For this data, access methods are provided to obtain the information on demand – automatically converting the data from the source semantics and format to the corresponding global representation. Other than the time delay in accessing



**Figure 2 DataFoundry Architecture**

remote data the user is unaware of which data is locally cached, and which is remotely accessed.

This architecture improves both base approaches in important ways. It improves the federated approach by increasing reliability and decreasing the cost of participating in a federation. By providing a single collection point for warehouse resources, the local store transfers the participation cost from the data sources to the warehouse. This is critical in an environment where individual data sources may not have resources to contribute to a federation. The consolidation into a single location also reduces network traffic when the amount of data returned is large, when there are joins in the query, or when the network connection slow. Finally, by acting as a cache for the data sources, DataFoundry can provide correct results when data sources are over-burdened or unavailable. This approach also improves upon traditional warehouses by maintaining access to detailed information while reducing storage costs by not duplicating non-critical data. If the identification of “critical” data is done properly, this approach ensures the warehouse remains useful to research scientists while reducing the impact of source schema changes. These benefits improve the overall system performance and decrease the investment required to create and maintain the warehouse.

Because data is cached locally, it must be regularly updated to reflect changes in the source. Currently, we are using simple batch processing techniques to automatically download and integrate the new data from each data source nightly. This technique, while primitive, is effective in an environment such as genomics where data is monotonically increasing and changes to existing data are rare. When modifications to either the schema or previously entered data are identified, the appropriate updates are explicitly made by the administrator.

## 5. Data Integration

Once a global schema has been identified, the data from the various sources needs to be converted into it. The traditional approach to data integration has been

to write a complex wrapper that parses data from the source, converts it into the target representation and passes it to the warehouse. Unfortunately, when the data source changes its representation, this large and complex program needs to be modified to properly interpret the new format. In addition, when a new source is added to the warehouse, a new program must be generated. Reducing the time required to modify and create the appropriate mediators and wrappers is crucial to the long-term feasibility of the warehouse. DataFoundry makes extensive use of meta-data to decrease the effort required to adapt to these changes by automatically generating an API and mediator. We have written a “compiler” (called the mediator generator, or MG) that translates declarative, highly-structured, meta-data into C++ classes.

The most important question regarding this meta-data is “What is the exact set of information needed to generate the mediators and API?” In order to answer this question, we need to look at what tasks the MG must perform in generating the mediator. Most of the mediator generation is straightforward parsing and code creation; however, the MG also needs to be able to resolve the type, format, structural, and semantic conflicts that arise between the data sources and the warehouse. In order to resolve type and format conflicts, the MG needs information about the various data concepts, their characteristics and data formats, (*abstractions*) and ways to convert between them (*transformations*). Resolving structural conflicts requires *mapping* information describing how the *abstractions* relate to the *database schema*. Finally, in order to resolve semantic conflicts, the MG uses information about the *abstractions*, *database schemata*, *mappings*, and legal *transformations*. Since this is all of the functionality required by the mediators, these four types of meta-data are sufficient for the MG to perform its task.

The MG uses the meta-data to define mediators that accept data from a wrapper through an API, transform the data appropriately, and enter it directly into the warehouse. To incorporate a new data source into the warehouse, the administrator first extends the abstraction definitions where appropriate, ensuring the required transformation methods are defined, then uses the MG to create the API and mediator. The administrator must also write a wrapper to read the data from the source and pass it through the API to the mediator. Because the wrapper is external to the MG, tools such as Lex/Yacc may be used to generate it. For example, when we integrated the SCoP taxonomy database, we defined warehouse tables to represent taxonomy information; extended the abstractions to include information about taxonomies; updated the mappings to the warehouse; generated a new mediator; and wrote a parser that reads the flat file representation of SCoP and passes the data to the mediator.

The use of meta-data to create the mediators will significantly reduce the effort required to maintain the warehouse in a dynamic environment – we have already demonstrated a significant reduction in the time required to integrate new sources into our prototype.

Adapting to minor schema changes often requires only modifying the parser to read the new format. Significant changes may require the meta-data to be modified and a new mediator created. Additional information about the meta-data and the MG can be found in [7][8].

## 6. Conclusions

The inability to fully utilize the wealth of publicly available information is a significant problem, not only for bioinformatics, but for scientific domains in general. While commercial products are currently available, they are focused on business applications and do not meet the unique needs of scientific domains. In particular, they do not address either the subtle data integration issues resulting from the complex relationships between scientific data, or the more obvious schema integration issues resulting from the dynamic nature of the sources.

DataFoundry addresses these problems by utilizing a mediated warehouse architecture to provide a consistent interface to scientific data. This architecture reduces the overall storage requirements of the warehouse, while maintaining access to all available data. Furthermore, our unique view of inter-database correspondences and extensive use of meta-data differentiate this approach from others while providing a significant reduction in maintenance costs. While DataFoundry has focused on the genomics domain, it presents a general-purpose approach to managing scientific data, allowing scientists to better utilize the data they have worked so hard to produce.

## Bibliography

- [1] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its new supplement TrEMBL. *Nucleic Acids Res.* 24:21-25(1996)
- [2] C. Batini and M. Lenzerini and S. B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration" *ACM Computing Surveys*, 1986 December, 18:4, pp 323-364
- [3] M.S. Boguski, T.M. Lowe, and C.M. Tolstoshev. "dbEST – database for expressed sequence tags". *Nat. Genet.* 4(4):332-3. Aug 1993.
- [4] M. W. Bright, A. R. Hurson, and Simon H. Pakzad. A taxonomy and current issues in multidatabase systems. *Computer*, 25(3):50-59, March 1992.
- [5] J. Callaway, M. Cummings, B. Deroski, P. Esposito, A. Forman, P. Langdon, M. Libeson, J. McCarthy, J. Sikora, D. Xue, E. Abola, F. Bernstein, N. Manning, J. Sussman. Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description Version 2.1. Tech Report available from [www.pdb.bnl.gov](http://www.pdb.bnl.gov). October. 1996.
- [6] A. Chen and V. Markowitz, An Overview of the Object Protocol Model (OPM) and the OPM Data Management Tools, *Information Systems*, Vol. 20, No. 5, 1995.
- [7] T. Critchlow, M.Ganesh, R. Musick, "Automatic Generation of Warehouse Mediators using an Ontology Engine". Proceedings of the 5<sup>th</sup> International Workshop on Knowledge Representation Meets Databases (KRDB). May 1998.
- [8] T. Critchlow, M.Ganesh, R. Musick. "Meta-Data Based Mediator Generation. Proceedings of the 3<sup>rd</sup> IFCIS Conference on Cooperative Information Systems (CoopIS). Aug 1998.
- [9] E.-P. Lim and J. Srivastava and S. Prabhakar and J. Richardson, "Entity Identification in Database Integration", in *Proceedings of Ninth Int'l Conference on Data Engineering*, pp 294-301, April 1993, Vienna, Austria
- [10] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and Jennifer Widom. Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. In *Proceedings of the AAAI Symposium on Information Gathering*, pp. 61-64, Stanford, California, March 1995.
- [11] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*. 1995. 247:536-540.
- [12] E. J. Neuhold, W. Kent, M.-C. Shan, "Object Identification in Interoperable Database Systems" in *Proc. of First Int'l Workshop on Interoperability in Multidatabase Systems(IMS91)*, IEEE Computer Society Press, pp 302-305, April 1991, Kyoto, Japan
- [13] G. C. Overton, S. B. Davidson, and P. Buneman, Database Transformations for Biological Applications *DOE HGP Contractor-Grantee Workshop VI*, Santa Fe,NM, Nov 97.
- [14] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed heterogeneous and autonomous databases. *ACM Computing Surveys*, 22(3):183-236, September 1990.
- [15] D.-G. Shin et. al., Graphical Ad hoc Query Interface for Federated Genome Database, Computer Sc. & Eng. U of Connecticut, *Storrs CT DOE HGP Contractor-Grantee Workshop VI*, Santa Fe,NM, Nov 1997

---

Work performed under the auspices of the U.S. DOE by LLNL under contract No. W-7405-ENG-48 UCRL-JC-133640