DataFoundry:

Warehousing Techniques for Dynamic Environments

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

Center for
Applied Scientific Computing

UCRL-JC-129576
January 1998
This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This is a preprint of a paper that was submitted to The Sixth International Conference on Intelligent Systems for Molecular Biology, Montreal, Quebec, Canada, June 28-July 1, 1998. This preprint is made available with the understanding that it will not be cited or reproduced without the permission of the authors.
DataFoundry:
Warehousing Techniques for Dynamic Environments

T. Critchlow**, K. Fidelis†, M. Ganesh‡, R. Musick*, T. Slezak‡‡
* Center for Applied Scientific Computing
† Biology and Biotechnology Research Program
‡ Computer Applications Organization
‡‡ Joint Genome Institute
Lawrence Livermore National Labs (LLNL)
P.O. Box 808, Livermore CA 94551
wk. (510) 423-5682 fax: (510) 423-2993
{critchlow | rmusick | ganesh | fidelis | slezak }@llnl.gov

Keywords: data warehouse, database integration, structural biology, homology, ontology

Abstract
Data warehouses and data marts have been successfully applied to a multitude of commercial business applications as tools for integrating and providing access to data located across an enterprise. Although the need for this capability is as vital in the scientific world as in the business domain, working warehouses in our community are scarce. A primary technical reason for this is that our understanding of the concepts being explored in an evolving scientific domain change constantly, leading to rapid changes in the data representation. When any database providing information to a warehouse changes its format, the warehouse must be updated to reflect these changes, or it will not function properly. The cost of maintaining a warehouse using traditional techniques in this environment is prohibitive. This paper describes ideas for dramatically reducing the amount of work that must be done to keep a warehouse up to date in a dynamic, scientific environment. The ideas are being applied in a prototype warehouse called DataFoundry. DataFoundry, currently in use by structural biologists at LLNL, will eventually support scientists at the Department of Energy’s Joint Genome Institute.

1. Introduction
To date, the Human Genome Project (HGP) has been very successful in increasing sequencing throughput to meet the rigorous demands of its schedule. In 1987, the year the HGP began, the resources existed to sequence on the order of 100,000 bases per year. Today, the genome community is hoping to identify and sequence on the order of 200,000,000 bases annually, and this number is projected to increase dramatically over the next couple of years. This improvement reflects a significant increase in sequence production facilities available to the HGP, as well as advances in sequencing technology. However, sequencing is only the first of several steps required to identify (and affect) the role and function of various genes in the human genome. Unfortunately, not all of the other steps in this process have shown the same success in improving throughput as has sequencing. In particular, while identifying the structure of a protein is an important step
in determining its function, it remains a non-trivial undertaking; both time consuming and costly. Current techniques and resources limit structural identification to only a few hundred unique proteins each year. The several order-of-magnitude gap between the rate of sequence and the rate of structure determination has significant implications. Bridging it requires more than just throwing additional resources at the problem - revolutionary breakthroughs in structural identification are desperately needed.

Advanced computer analysis is one of the most promising avenues for resolving this gap. In particular, protein fold recognition and comparative modeling hold out hope that a better understanding of the relationships between sequence patterns and protein structure, of protein structure itself, and of protein function and associated chemical reactions, will make structure prediction via these methods feasible. Because full-scale physics modeling is currently impractical, this knowledge must be obtained by careful examination of relevant information including known structures of proteins, folding properties and propensities, protein and DNA sequences, and genetic organization and expression data, to name a few. Much of this information is currently available on the Web, but is spread across multiple community databases, each with its own particular set of concepts, semantics, data formats, and access methods. The burden falls on the scientist to resolve these conflicts, integrate the data, and interpret the results. More often than not, these barriers prove too difficult to overcome, and data becomes an expensive but under-exploited asset. Providing cost effective access to data located throughout a large community is not a just a problem for structural biologists, or even the genome community as a whole, but has become a central concern for any modern data-driven enterprise.

DataFoundry is a data warehousing project that directly addresses this issue. The goal of the project is to provide structural biologists at LLNL a uniform and semantically consistent interface to data from multiple external and internal sources. The interface is Netscape-based, and allows both form-based and *ad hoc* SQL queries over a single global schema which represents information from all incorporated sources. To date, Protein Data Bank (PDB) [4] and SWISS-PROT [1] data have been integrated into this framework. Eventually, additional sources supporting various Joint Genome Institute (JGI) bioinformatics needs will be incorporated. Beyond being a component of the information infrastructure for the JGI, DataFoundry is a computer science research effort that explores various techniques geared to make warehousing a cost effective and feasible option in highly dynamic environments. The project is distinguishable from other efforts in the bioinformatics community, and the commercial data warehousing world, by four salient points:

1) A *practical resource* - We are not building a new warehousing tool, but rather using and extending available tools to build an actual warehouse supporting day-to-day biology research. A prototype is currently in use by structural biologists at LLNL.
2) A *consistent view of the data* - All of the underlying semantic and syntactic conflicts between the data sources are resolved, removing this burden from the user.
3) *Semi-automatic schema evolution and integration management* - A major research goal for this project is to significantly reduce the effort required to create and maintain a warehouse that integrates rapidly evolving data sources. This problem has been encountered in most scientific data management efforts, but techniques for managing change are still primarily manual in nature. DataFoundry is working on two approaches (described in depth below) that should make progress towards
automating this process: natural keys for relating data, and a significant ontology infrastructure which incorporates meta-data into every step of the integration process.

4) **Extended homology and intelligent search** - Sequence homology searches are frequently used to identify related proteins. Homology services on the Web commonly allow a user to enter a sequence, and will eventually send an email response containing the results of the sequence homology search on the selected data. We are extending this functionality in two important ways. First, we are precomputing sequence homologies for data from both PDB and SWISS-PROT using multiple parameter settings for several standard sequence comparison algorithms. An SQL interface allows complex searches over these scores, permitting new and interesting questions to be formulated. Second, beyond traditional sequence homology, DataFoundry allows construction of generic search filters that can be applied to the integrated data set. These filters could include sequence homology scores, structural homology scores, functional homologies, or in general any description of a cluster expressible in SQL over the DataFoundry global schema.

This paper describes DataFoundry and focuses on research geared at automating the schema integration and evolution tasks. While there are several other challenges that must be addressed in any data warehouse venture, such as data cleaning and consistency, we do not have sufficient room to explore these issues here. We begin with a brief overview of related efforts in the next section. Section 3 describes our high level approach to integrating the data sources, while Section 4 concentrates on how we identify and represent inter-database correspondences. The implications of our meta-data approach are briefly discussed in Section 5; the details will be presented in a forthcoming publication. Section 6 describes the current status of the project, and outlines future directions, and finally, we conclude with a brief summary.

2. **Background**

Data management issues in the business information-processing domain form a well-known subject area and technological advances in the field have been primarily driven by the requirements of this domain. Relational database technology is accepted as a de-facto standard for transactional systems and several vendors provide mature database management systems based on this technology. SQL is a well-known data access and manipulation language available across most of these platforms. Along with the accumulation of large volumes of data, came newer requirements for utilizing the information contained therein intelligently. Data warehousing technology has been projected as the solution for analytical processing needs in information management. Currently, in the business applications domain several commercial tools exist to address the various steps involved in creating and maintaining a warehouse in this environment.

Scientific applications, on the other hand, face a different set of problems. Existing information sources do not have a widely accepted format for storage and management of resources; some rely exclusively on flat files while some utilize 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
the bio-informatics domain and compare them with the approach taken by the DataFoundry project.

The Object Protocol Model (OPM) [5] project at Lawrence Berkeley National Laboratory provides a set of tools for data management, focusing primarily on applications in the scientific domain. This project recognizes that representing scientific experiments requires more complex data models and modeling facilities than are provided by commercial database management systems (DBMSs). Thus the OPM data management tools use an object-oriented model for data, and have included a protocol class for modeling experiments. The OPM toolset consists of a schema editor for specifying and managing schemas in OPM, a graphical querying and browsing tool, and tools for conversion between relational and OPM schema definitions to work with existing scientific database. The OPM toolset has also 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, and may be used within DataFoundry to remove some of these obstacles.

The CPL/Kleisli [10] 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 the use of 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) [7] 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 worldwide web. The usefulness of this approach in a scientific application will depend greatly on the ease with the data in the particular domain can be represented using OEM.

The Graphical Query Interface project for Federated Genome Databases [12] at the University of Connecticut 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 schemas, which will be executed at the corresponding sites. Meta-data information 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 to the federation requires updating all inter-database relationships.
DataFoundry combines many of the advantages of the systems described above to form a unique approach to database integration. It uses an integrated global schema to present a consistent view of the data. An ontology captures the meta-data required to map between the source schemata and the global schema, significantly reducing the effort required to integrate a new data source. Users pose queries against the global schema and the translation to queries against source schemata is automatic. In addition, representing a subset of the data locally reduces network traffic and provides a significant performance improvement over completely distributed approaches.

3. The DataFoundry Architecture

Traditionally, three technologies 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 [11] define an integrated schema over the subset of data that is both interesting to the federation as a whole, and available from the data sources. This global schema represents a virtual database containing data from each participating source within a single, consistent representation. While the global schema may incorporate only a portion of the data available from the data sources, information not explicitly contained in at least one participating data source cannot be represented. 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. 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. Additionally, because data is not represented locally, this approach is susceptible to long delays when answering queries, and furthermore, may return misleading or incorrect results when a data source is temporarily unavailable.

To reduce the amount of network traffic and improve query results, data from each data source can be incorporated into a single database – resulting in the traditional, monolithic data warehouse. There are two problems with this approach. First is the tremendous amount of storage required to keep all of the data, in some cases several terabytes worth, within a single database. To reduce storage requirements, many warehouses contain only summary and aggregate data. Obviously, warehouses using this data transformation provide very different types of information than the original data sources, and may be of limited use in a scientific environment. Second, is the issue of keeping the data current and the warehouse fully functional. In some application domains, such as accounting, only a small fraction of the data changes at any given time, and previously entered data is not expected to change. More importantly, the source
schemata are quite stable (e.g. the notion of an account rarely changes). In dynamic scientific areas such as genetics, a large portion of the data changes on a regular basis. Existing entries may change based on new information, and source schemata are rapidly evolving with our understanding of the underlying biology.

DataFoundry is based on a mediated data warehouse architecture [7], which is a combination of the monolithic warehouse and federated database approaches. The initial schema used by this architecture is a global schema, similar to a federated database, which contains a subset of the information in the data sources. This schema is then expanded to include some summary and aggregation information. A local data store is used to represent this additional data, as well as replicating both the most important and the most frequently accessed data. The result is a consistent view of the data and greatly improved query performance, without the same storage costs as a monolithic warehouse. Queries involving data not represented in the local store require accessing a remote site, as in a federated database. As each data source is incorporated into the warehouse, the subset of critical data is identified and represented on the local store. For data considered non-critical, data access methods are provided to obtain the information on demand, automatically converting the data from the source semantics and format to the corresponding warehouse representation. Other than the time delay in accessing 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 allowing most queries to be answered quickly and correctly using local resources only. While reducing network traffic is important when the amount of data returned is large or the network connection slow, the most significant benefit is the ability to provide correct results in the presence of over-burdened or unavailable data sources. Additionally, by providing a single collection point for warehouse resources, the local store moves the cost of participating 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. It improves the traditional warehouse approach by maintaining access to detailed information while 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.

Unfortunately, because data is being cached locally, the issue of data currency needs to be addressed. We expect to pursue an agent-based approach to identifying and incorporating modified data and to use the ontology engine to automate several of the steps required to incorporate data and schema changes over the remainder of the project. Currently, however, we are using simple batch processing techniques to automatically download and integrate the new data from each data source nightly. Modifications to either the schema or previously entered data are manually identified, and the appropriate updates are explicitly made by the systems administrator.

4. Integrating Heterogeneous Data Sources

Providing access to data from various sources is a critical step in developing a warehouse. For the warehouse to be useable, the data should be presented in a single, coherent, global schema. 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 differences that occur between the individual representations. Traditionally, a global schema is created by conceptually overlaying the source schemata, with similar concepts and attributes being 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) [6][9] or 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 data, 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 a common occurrence due to the high level of heterogeneity present in this domain.

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 database; otherwise they will remain distinct. Neither choice is entirely acceptable. If the instances are combined, the resulting representation will select one of the original sequences to represent it. 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 resolved 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.

**4.1. Natural Keys**

Instead of overlaying complex concepts, primitive concepts likely to be represented within multiple data sources are identified and inter-database relationships permitted between these concepts only. 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 most natural keys can be described as a single attribute that is associated with all representations of a more complex concept. Furthermore, the semantics of each representation will be closely related and easily generalized to a single, inclusive semantic definition.

To demonstrate the process of identifying a natural key, consider the PDB. As the official repository for molecular structure data, PDB associates a given protein sequence with its 3-D atomic structure. Several other concepts are also related to the protein including, publications, secondary structure information, the names of people who performed the experiment, etc. Upon close examination, it becomes apparent that there are four concepts likely to be shared with other databases: the sequences, publications, protein names, and people associated with the experiment. The other concepts, for example the coordinates and the entire PDB entry, provide information at the wrong level of detail. The coordinates are too primitive for inter-database queries, while the PDB entries are too complex and would create undesired relationships.

Care must be taken to ensure that the semantics associated with natural keys are sufficiently general to permit integration with other data sources. Continuing our example, if the concept of a sequence were restricted to that of a PDB chain, integrating other databases, such as SWISS-PROT, would be difficult. However, by redefining our concept of sequence to be “an arbitrary sequence of amino acids” related concepts can be identified within multiple data sources. Further, while the physical representation of the concept may differ between the data sources (e.g. a sequence may use one character or three character amino acid codes) there will always be a trivial mapping to this fundamental concept. Thus, the sequence forms a natural key for the PDB database. A similar analysis confirms that protein names, publications and people also correspond to natural keys.
4.2. Inter-Database Correspondences

We have identified three important categories of inter-database relations: identity, similarity, and traditional relationships. We describe the first in detail in the remainder of this sub-Section, and the second in Section 4.3. Standard 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, or more precisely the almost-identical, relationship relates those instances of natural keys in different databases that appear to represent the same object. This relationship is an explicit representation of the implicit relationship described in the previous section. 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 structures represented in Figure 1(a); in our approach the global schema would contain both original structures related through this relationship on both the sequence and name attributes. The heterogeneity present in the sources remains, but the correspondence between the tissue and the function is also identified. To provide even more flexibility, we can use alternative methods for defining correspondence, such as high homology scores, as opposed to the default identity function.

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 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.

4.3. Extending the Concept of Similarity

The similarity relation identifies correspondences between instances of common objects that are similar in some sense. Our current emphasis is on representing sequence homologies between proteins using results from the common homology programs, including BLAST, FASTA, and Smith-Waterman, at several different settings. In the future, the similarity relation will be extended to include other algorithms, such as one for structure based homology being developed locally. While dynamically computing individual homology scores for a particular sequence is straightforward, a repository of pre-computed homologies has several advantages. First, it provides a batch interface significantly reducing the effort required to obtain the results of multiple homology queries at the same time. Second, it is readily searchable permitting new and interesting questions to be easily formulated. For example, primitive clusters can be obtained by identifying the transitive closure of homologous proteins. Finally, this repository is a valuable resource for data mining research we intend to pursue.

---

1 Relationships that occur solely within a single database are trivially included within the global schema when the enclosing source schema is integrated.
The primary obstacle in creating this relation is the large amount of computer resources, both storage and computational, required to populate it. For the sequence homologies alone, we expect to record on the order of 100 million entries. Because of the significant computer resources required to even compute all of these values, we are making extensive use of the supercomputing facilities available at LLNL. A second obstruction is a theoretical problem associated with storing homology scores: many are probabilistic values based on the data available at the time of execution, and thus become obsolete over time. We are taking a two-pronged approach to this problem. First, whenever a score is presented to the user, a confidence in the score is also presented. The user will have the option of dynamically recomputing the score if the confidence is too low. Second, we plan to use the observation that the scores do not change quickly once sufficient data are available to spread the recomputation over a large time frame. After a statistically determined percentage of the data has changed, a low priority process will recompute the scores for all of the data in the system based on the new data. We believe that there will be sufficient resources to recompute these values before the recorded values become unrepresentative.

5. Ontology

A major challenge in maintaining a warehouse in a dynamic field such as genetics is quickly adapting to schema changes in the data sources. Consider a warehouse that integrates 12 different data sources, each having two schema revisions per year. In this case, the warehouse needs to respond to a source schema modification about every two weeks. Understanding the changes and modifying the mediators to accept the new schema may easily take two to three person weeks if done by brute force, during which the warehouse may not be fully operational. Reducing the time required to incorporate these schema changes is crucial to the long-term feasibility of the warehouse. Our approach makes extensive use of meta-data to reduce the effort required to adapt to these changes.

Our meta-data is represented in a single ontology containing abstractions of domain specific concepts, descriptions of the data sources and the warehouse, and mappings between them. Domain specific abstractions represent the aggregate of all information known about a particular concept. They are defined as a collection of characteristics representing various aspects of the concept. Each characteristic is comprised of attributes that correspond to different representations of the same primitive concept. For example, the abstraction chain has a characteristic representing its sequence, and that characteristic has attributes for different representations (e.g. a string comprised of one-character amino acid codes, and a string comprised of three-character amino acid codes separated by spaces). An attribute may be specified as the default for a characteristic, and transformation functions are defined between it and the other attributes. For each database, the concepts it represents are described as a set of C++ classes. There is a direct mapping between the attributes of these classes and attributes of the corresponding abstractions.

The ontology is used to define mediators that accept data from a parser through an API, perform the appropriate transformations, and incorporate it directly into the warehouse. To incorporate a new data source into the warehouse, the warehouse administrator must define the appropriate set of C++ classes, map each source attribute to
the corresponding attribute in the ontology, and ensure that the required transformation methods are defined. Then the ontology engine creates the API and mediator methods based on the information specified in the ontology. The administrator is still required to write a parser to read the data source and make the appropriate API calls. However, because the parser is external to the ontology engine, it may be any program that can interact with the mediator; other tools, such as Lex/Yacc, may be used to generate it. Adapting to minor schema changes often requires only modifying the parser to read the new format. Significant changes in the data being represented may require the ontology to be modified and a new mediator created.

We believe the use of meta-data to create the mediators will significantly reduce the effort required to maintain the warehouse in a dynamic environment. We are also planning on developing a graphical interface to the ontology, further reducing the effort required to create the initial transformation.

6. Next Steps

Over the next year, the ideas laid out above will be implemented in DataFoundry. The current Netscape-based interface will be extended, with graphical schema browsing and query editing capabilities provided by either OPM or the University of Connecticut project. By the end of the year, we expect to have integrated all the databases required to support LLNL biologists in their modeling effort, and begun integrating the sequencing information required by the JGI. This involves incorporating the SCOP [8] database as well as genome physical map data. On the research end, extending DataFoundry capabilities in automated schema integration and evolution remains paramount. Applying and extending current object identity techniques with the use of data mining will become more of a focus for the project. Finally, DataFoundry will be used as the primary data resource for domain-specific exploratory data mining efforts. Conceptual clustering and pattern recognition techniques will be applied to the protein folding problem. Clustering will be carried out both at the protein level, and the sub-protein level (domains, chains) based on homology information, taxonomy, and various position and residue-specific metrics.

7. Conclusions

The inability of structural biologists to determine the structure of proteins at the rate required by the sequencing and functional analysis components of the HGP is becoming a significant problem. If new technology that significantly reduces the time necessary to structurally and functionally characterize a protein is not introduced in the next few years, the success of the project may be in jeopardy. One of the most promising approaches is the use of computer algorithms to derive structural and functional characteristics by analyzing DNA sequence. However, in order to make such inferences, the program must have access to a vast array of information that is currently distributed across several databases.

The DataFoundry project addresses this problem by utilizing a mediated warehouse architecture to provide a consistent interface to these databases. The DataFoundry approach reduces the overall storage requirements of the warehouse, while maintaining access to all available data. Our unique view of inter-database correspondences and extensive use of meta-information differentiate this approach from
previous attempts to integrate genetics databases. These differences provide a significant reduction in the effort required to maintain a warehouse in a dynamic environment such as genetics. In addition, by defining and populating the similarity relation, we are providing a unique and important resource to aid the genomic community in future data mining efforts. While the potential uses of DataFoundry are many, its main benefit is to reduce the overall cost to the biologist of using the data that the HGP community has worked to hard to produce.

Bibliography


