RI 02018 29 April 2002 Computer Science

IBM Research Report

Data Integration Approaches in Bioinformatics - A Tutorial Survey

Biplav Srivastava
IBM Research Division
IBM India Research Lab
Block I, I.I.T. Campus, Hauz Khas
New Delhi - 110016. India.

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

Data Integration Approaches in Bioinformatics - A Tutorial Survey

Biplav Srivastava

Email: sbiplav@in.ibm.com
IBM India Research Laboratory,
Block I, IIT Campus, New Delhi 110016, India.

Abstract

The amount of genomic data available for analysis online is vast and ever growing. Yet, a biologist wishing to gain insight from them is lost in data model, data formats, and interfaces of particular data sources. In this article, we survey the architecture of data integration (also known as information gathering or informatics) applications which are used to integrate and provide uniform access to heterogeneous data sources in bioinformatics. We use a taxonomy to highlight the capabilities of recent integrated systems for genomic data access and give pointers to their future extensions.

1 Introduction

The amount of genomic data available for analysis online is vast and ever growing. Yet, a biologist wishing to gain insight from them is lost in data model, data formats, and interfaces of particular data sources. Many sources have data as formatted file with specialized Graphical User Interfaces (GUIs) and retrieval packages. The design choices made by the autonomous data sources considers the complexity of data, efficiency of analytical tools, multi-platform support and cost of implementation, but not integration issues which would have lead to more adoption of database management systems. The heterogeneity of the data sources and the multitude of non-standard implementations present an integration nightmare for uniform access.

The computer science field of data integration[19] (also known as information integration or information gathering[18]), which lies at the cross-roads of Artificial Intelligence and Databases, studies how to provide access to multiple autonomous heterogeous data sources in a uniform fashion[19] such that the diversity and distributed nature of the data sources is replaced by the

virtual model of data accessible to the end user. We further define access as the ability to model, retrieve and query the individual or collective data from the data sources in a consistent manner. Given a global world model, a set of information sources, a mapping of contents of information sources to the world model, and a query on the world model, the objective is to return information contained in the information sources that answers the query on the world model. An agent integrating the different heterogeneous sources must return only the actual information that satisfies the user's query and no more. A specific example is a biological [1] application to model, query, and annotate biochip experimental data.

1.1 Bioinformatics: A complex data integration domain

We are interested in data integration in the large and complex domain of bioinformatics to facilitate data analysis. Here, according to a recent survey[4], there are atleast 335 data sources in 2002 beginning [2] with at least 6-10 sources of similar type (for example, protein, pathway, publication, gene expression data, etc.). In contrast, conventional data integration solutions have dealt with very few sources with little overlap in content. Moreover, there is rich domain information on how results for queries should be obtained and strong user preference for sources (example, one biologist may prefer SWISS-PROT to PIR for protein information due to its affiliation). The completeness (but not correctness) of the results is negotiable in favor of performance and timeliness. The data size can be large (in megabytes or gigabytes). Finally, the user may want to employ the unique native data analysis capabilities of the data sources.

Data Analysis in Bio-informatics is in two phases:

(1) Identify genes which constitute an interesting regulatory pattern by applying a set of statistical analysis/ clustering methods like hierarchical, k-means, fuzzy and self-organizing feature maps.
(2) Identify functional relationships among selected genes based on maximum number of information/ data sources to develop and verify hypotheses. In a sense, the clusters from the first step are characterized by a set of meaningful features from the databases. Additionally, relationship among the genes is predicted and validated/understood.

Karp [14] has identified the issues in data integration in bioinformatics and alluded to the need of a reasoning module through what he calls a Relevance

[1] We are particularly interested in Bioinformatics, which is the application of information sciences (mathematics, statistics and computer science) to increase our understanding of biology.
[2] Up from 281 in 2001.

Module. An example of a query that the biologist may want to ask in *e2e* is (modified slightly from [14]):

Q: *Find examples of co-expressed genes that code for enzymes in a single metabolic pathway.*

Query Q can be more precisely described as - find a set of genes G from a pathway P where:

- There exists R, the set of all reactions in P
- There exists E, the set of all enzymes that catalyze a reaction in R
- G is the set of genes that encode an enzyme in E, and
- Genes in G are similar in their expression level according to some clustering algorithm.

To answer the query, the system needs to access a protein pathway data source like KEGG, the user's gene expression data and a clustering algorithm. We will use this query throughout the paper to highlight the working of the surveyed bioinformatics systems.

1.2 Data Integration

A data integration system can be characterized by the amount of transparency it provides to the user. The different types of transparency, in increasing degree of abstaction are:

(1) Format transparency or the user not having to know about the data formats supported by a source and the individual ways to access them.
(2) Location Transparency or the user not having to know about the location of a piece of information on the source. This gives the impression of a single location for all information while the access details to the source is handled by the system.
(3) Schema Transparency or the user not having to know the schemas of individual sources to reconcile the final result. This allows the user to query data across sources in an integrated manner.
(4) Source Transparency or the user not having to know if a particular source exists. For example, the user will know that protein information can be obtained from the system but she will not have to know the source from which such an information can be obtained. Source transparency necessitates a domain ontology so that the user interacts with the system using domain concepts and the concepts are in turn reconciled with available sources.

Conventional data integration approaches consider the data sources as repositories of data but not as applications (which may in turn embody complex interactions with other sources). They also do not provide direct mechanisms for leveraging domain-specific user guidance. A characteristic of bioinformatics is that since the number of sources on any particular subject is high, the data integration solution should be scalable with sources.

A variety of approaches have been developed for integrated access to heterogeneous data sources in genomics. In the *link-driven federation* approach, the user can switch between sources using system-provided links. The systems do not provide any transparency to the user. In *view integration*, a virtual global schema in a common data model is created using the source description. Queries on the common data model is then automatically reformulated to source level queries. A variation of view integration is the *warehousing approach* where instantiation of the global schema is created, i.e., all data of interest is locally stored and maintained for integrated access. Both view and data integration provide format, location and schema transparency to the user. The holy-grail in data integration is to provide source transparency as well, which leads to *semantic integration*. Here, a common ontology describes the concepts of interest and source characteristics are directly mapped to the common concepts. The user interacts in the ontological realm while the system deals with the sources.

1.3 Outline

In subsequent sections, we survey these approaches for data integration in detail in and give examples of typical implementations. As we find, though much stride has been made in integrated genomic data access, no single approach can handle the idealized scenario. We give pointers to ongoing research efforts and future trends.

2 Technologies in Data Integration

A basic problem underlying view integration is the *autonomy* of the sources, which invariably leads to the lack of cooperation and standardization of formats among the sources. Existing formats at major biological sites are EMBL and ASN.1 but they are biology specific and not universally adopted even in biology. If sites were to use IT industry-wide standard formats in biology, system developers could leverage the available tools, and thereby, ensure reliability and lower cost of system developement.

In the 90's, Internet and HyperText Markup Language (HTML) have become the dominant medium for publishing and sharing data. Hence, it is not surprising that a number of low-cost solutions to data-integration have followed HTML's paradigm of linking-up documents from different sources. The user can browse the documents through a standard viewer and navigate across sources by invoking available links.

The main technologies for facilitating universal communication among distributed components are OMG's Common Object Request Broker Architecture (COBRA[21]) and W3C's eXtensible Markup Language (XML[27]). CORBA specifies the standards that an object component should implement so that it is recognized by artifacts known as brokers which route client requests for service. The Life Sciences Research Task Force (LSR) within OMG has finalized specification for biomolecular sequence analyses and genome maps but individual sources have to provide an interface conforming with the specification.

XML has emerged as the de facto format for exchanging data in the last few years with abundant development tools and backing from major IT vendors. XML is ideal for format integration because it can handle semi-structured data and hence, is more felxible than relational data model where data must be structured, i.e, the attributes of an entity and their types must be known in advance. But XML is still just a data format and the participants of any collaboration must still agree on semantics.

Structure of data is specified in XML by Data Type Definitions (DTDs) while type and semantics can be specified by the upcoming XMLSchema. For semantic integration, the data sources in biology and the integration systems should talk about the common concepts in the same language. Ontologies are formal models which define concepts and their inter-relationships so that automated reasoning is possible. Work on ontologies would eventually lead to the standardization of XML tags and their well understood meanings.

3 Link Driven Federation

The link driven federation is a *hypermedia* environment. Here, a user starts from some point of interest in a data source and then can jump to other related sata sources through system created links. The user has to still interact with individual sources; only the interaction is easier through convenient links and not invoking the sources directly. SRS[9], GeneCards[23] and LinkDB[10] are examples of this approach. A similar system, but not based on links, is described in (cite patent) where the the middleware provides a set of procedural choices to the user and lets user interaction determine how the data is fetched.

The link driven approach is very convenient for non-expert users because of the simple point-and-click user interface. It is also possible to perform limited keyword search on the content of a source by specifying regular expressions.

The downside of link driven approaches is that it does not scale well and has no across-source capabilities. When a new data source has to be added into the system, links connecting its entries and that of all the other data sources has to be created. If a data source changes, the link building has to be re-done. Moreover, a join between the data of two sources is not possible in link driven approaches.

Recall that query Q involves protein pathways data, gene expression data and a clustering algorithm. If the user wanted to run query Q in a link driven system, she would have to *manually* perform the following steps:

(1) Retrieve gene expression data and find clusters of co-expressed genes. For each cluster C, manually perform the following:
(2) Access a pathways resource. For each pathway P, manually check the following:
(3) Find R, the set of all reactions in P.
(4) Find E, the set of all enzymes that catalyze a reaction in R.
(5) Let G be is the set of genes that encode an enzyme in E. If G is contained in C, C is a solution.

A link driven federated system helps the user by facilitating source access through conveniently inserted links but the user has to explore the query plan search space manually.

3.1 SRS

The creators of the Swiss-Prot database at the Swiss Institute of Bioinformatics and the European Bioinformatics Institute have created SRS (Sequence Retrieval System)[9]. SRS allows retrieval from an extensive catalog of more than 75 public biological databases of interest. The link button in SRS allows the user to obtain all the entries in one databanks that are linked to an entry (or entries) in another databank. Hyperlinks are links between entries which are displayed as hypertext (clicking on the text takes you to the related entry). These are hardcoded into SRS and are useful for examining entries that are referenced directly from a data item of interest.

3.2 LinkDB

The integrated database retrieval system DBGET/LinkDB[10] is the backbone of the Japanese GenomeNet service. DBGET is used to search and extract entries from a wide range of molecular biology databases while LinkDB is used to search and compute links between entries in different databases. Once an entry is retrieved through DBGET, all links from this entry can be obtained by clicking on the entry name, which causes the search against LinkDB. In addition to the original links provided by the source database which are embedded in the entry, LinkDB also aims at providing computer-generated links, which include:

- Factual Links: links between database entries, e.g., Medline ID and GenBank accession
- Similarity Links: links produced by similarity search, e.g., the results of BLAST and FASTA
- Biological Links: links by biological meanings, e.g., molecular or genetic interactions in the KEGG pathways.

4 View Integration

In a typical view integration system, a request from the user is take by the client application and passed on to the middle level. The middle level analyzes the request, transforms it into a number of executable sub-requests and passes them to wrappers. Wrappers are source-specific modules that mask the peculiarities of the source with respect to source access and format of results returned. A wrapper will suitably transform the sub-request and interact with the data source for the response. The response is transformed into a format that the middle-level can handle and returned back. The responses is further processed by the middle level depending upon the original request and the final result is returned to the client application.

Many researchers have noted that if the middle-level can encapsulate a data model for the common available data, the user of a data integration system can use the query mechanism on the data model to access the virtual store, regardless of the data model of the individual sources. Another benefit is that since query languages are declarative, the user can be isolated from the internal mechanism of data retrieval allowing the latter to be independently optimized.

The capability of a view integration architecture is primarily determined by its middle level (middleware). View integration systems build a global schema from the schemas of heterogeneous data sources, and expose the global schema

to the user for querying. There are two approaches - *global-as-view* where the global schema is defined as a view over the local sources, and *local-as-view* where a global schema is defined beforehand and the local sources are described as views over the global schema.

Internally, middleware is then responsible for four functions:

- Query decomposition and query planning
- Query execution
- Data retrieval from wrappers
- Post-processing and presentation to user.

The advantages of the view integration approach are that the latest content of the sources is always returned by the system, operations across sources can be specified in the query language, no storage is needed at the middleware, and adding new data sources is easier than link driven approach.

The disadvantage is that since sources are accessed instantaneously during query execution, in the event of a source being down, a query can return less matches or even fail. Also, to build the common integrated view, deep expertise of systems is needed.

If the user wants to run query Q in a view integration system, she would have to perform the following steps:

(1) Specify the query in a declarative query language.
(2) Analyze the results produced.

A view integration system helps the user by requiring the query to be only specified. The system takes the responsibility of exploring the query plan search space to return the results. However, since the specification of the query is dependent on the data model used for integration, it can be non-intuitive and quite complex.

4.1 Discovery Link

DiscoveryLink[12],[15] is an IBM offering for information retrieval based on a relational data model for the virtual collection of data. The user views Discovery Link as a uniform relational database system with a global schema. This allows the user of DiscoveryLink to use SQL to query the heterogeneous sources regardless of whether the sources had the relational data model or others (e.g., text, XML). Figure 1 illustrates the DiscoveryLink architecture.

In Discovery Link, the capability of the data source is recorded through a

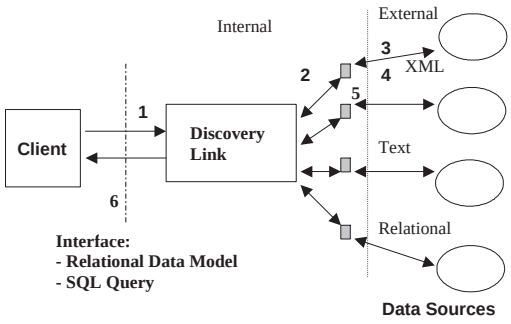

Fig. 1. Discovery Link Architecture.

wrapper and data source registration process. A source records following information to Discovery Link:

- The data schema of interest from the source
- SQL capability of the source (example, can it handle test on predicates)
- Data model, cost and initialization parameters

Discovery Link will use the registration information for query planning and optimization of query execution.

When a SQL query is given, Discovery Link's query subsystem figures out the sources that have to be invoked based on their registration information, and the corresponding wrappers. The initial query is decomposed into sub-queries which are dispatched to the wrappers. It is the job of the wrapper to retrieve the necessary data and perform any transformation over the result, so that the result is relational. Finally, the results from the sub-queries are combined and presented to the user.

4.2 Kleisli/K2

K2[8], and its predecessor, Kleisli [6], are view integration systems from University of Pennsylvania. K2 and Kleisli are based on a data model of complex values where collection types, i.e., lists, sets and bags, support type union and may have arbitrary nesting. The Query language in Kleisli is Collection Programming Language (CPL) which is designed for manipulating and transforming complex data and matches the expressivity of Structured Query Language (SQL) when restricted to relational data, but has different syntax. K2 supports Object Query Language (OQL[7]) extended with disjoint union types and has SQL's "select-from-where" syntax.

Like DiscoveryLink, K2[3] uses individual data drivers or wrappers to handle

[3] Though we focus only on K2 here, the description is also applicable to Kleisli

low level integration issues with each class of data sources. Once a query is issued in OQL, it is type checked based on source metadata, and decomposed into sub-queries that can be handled by the data sources. Query decomposition and optimization is handled by a rule-based optimizer which eliminates intermediate results and sends the biggest derived sub-queries to the sources. Any part of the query that cannot be delegated to the sources has to be executed by K2.

Since K2 is based on OQL, integrated views can not only be specified by view functions (as in Kleisli or DiscoveryLink) but also as user-defined classes at different abstraction levels. K2MDL, the class specification language, lets user describe classes based on how their attributes and extents can be computed from the available sources. View classes can be queried using OQL, and the optimizer can decompose them into suitable multisource queries using the class's K2MDL description.

K2 is a multi-threaded server application in Java that any client can communicate with using RMI-IIOP or socket protocols. Kleisli is implemented in Standard ML. K2 and Kleisli have wrappers for GenBank/EMBL/DDBJ, dbEST, SWISS-PROT, KEGG, EcoCyc, GDB, SRS-indexed databases and BLAST.

5 Data Warehousing

The warehousing approach can be described as view integration where the global schema is *materialized* i.e., an instance is created locally. The data from the different sources is downloaded, cleaned of erroneous entries, categorized into meaningful structures (manually or automatically curated) and formatted for suitable analysis. Usually, the instance is stored in a database (e.g., relational database) and can be queried with a database query language (e.g., SQL).

The biggest advantage of a warehouse for data integration is that the downloaded source data can be manipulated into suitable formats and annotated to facilitate integration and analyses. Execution of queries is usually very fast because all data is locally available and the system is reliable because there is no outside dependecy.

On the other hand, maintenance cost in warehousing is high because the downloaded data from the sources has to be kept fresh (i.e, synchronized with the online source). If this is not done, the result of a query may be computed on

with the differences noted above.

stale local data, thereby compromising result correctness. Also, there is a large initial cost and ongoing storage requirement.

5.1 GUS - Genomics Unified Schema

Genomics Unified Schema (GUS[8]) is a warehousing based data integration systems from University of Pennsylvania. GUS uses a relational data model and stores nucleotide, amino acid sequences and annotations in tables. The data sources already included are GenBank/EMBL/DDBJ, dbEST and SWISS-PROT. The straight forward advantages of GUS is availability clean data with custom annotations, and fast and reliable query performance. Additionally, one can access data in more intuitive and insightful ways than the sources allow because data in GUS is structured along ontologies of biological terms.

GUS builds and maintains a map between DNA sequence based entries at some sites and gene-based entries at others through its local storage of the necessary date. Its tables hold the conceptual entities that DNA sequences and annotations indirectly represent, which are genes, the RNA derived from these genes and the proteins from those RNAs. While transforming the data into gene-centric organization, it cleans data to identify erroneous annotations and mis-identified sequences. Then, it uses ontologies to structure the annotations for different organisms.

GUS facilitates data maintenance by tracking how data is generated/ accessed from sources and subsequently modified. This helps in learning about continuous changes (external or internal) to the knowledge of genes that data items represent. The revisions of original data can be by the source itself, annotations are slowly experimentally verified, and predicted values become more accurate with better algorithms. For example, with computationally-derived annotations, GUS stores the algorithm used, its implementation version, input parameters and the run time information.

To keep its database synchronized with external data sources after initial download, GUS retrieves updates and new entries from them based on the source's change schedule or periodically. The changed fields of the modified entries are detected from the database based on a difference operation, and updated accordingly. Both the new and updated entries are subjected to an annotation update process in which the protein and DNA sequence and their annotation, are transformed into gene and protein based entries. The user always sees a production version of the database while updates are made to the next development version. When the development cycle is over, the database version is put into production, and a new development version is

created.

6 Semantic Integration

Ease in data access to remote sources is necessary for a flexible integration systems but raw data in itself does not mean much. Usually, both data and its biological context determines the complete meaning (semantics) of an item. Semantics plays a key role in subsequently processing the data and interpreting the results. We discuss efforts in biology in building consensus on data formats and semantics.

If the user wants to run query Q in a semantically integrated system, the query specification will be intuitive and in the relevant concepts from the domain. The system may internally map the specification to a query form that is more amenable for automatic query decomposition based on the (data) integration infrastructure. The user would perform the following steps:

(1) Specify the query in terms of domain concepts.
(2) Analyze the results produced.

6.1 TAMBIS

TAMBIS[11] (Transparent Access to Multiple Bioinformatics Information Sources) is an integration system for molecular biology where a domain-dependent ontology is used for source-transparent information retrieval. The biologist user interacts with TAMBIS at the semantic level of TAMBIS Ontology[3] (TaO, as it is called) while the system handles the source level interaction. TAMBIS uses the GRAIL language for description logic to build the domain ontology and leverages the middleware of BioKleisi, a variant of Kleisli[6] to implement data integration.

An ontology is a formal specification of concepts in a domain so that logical reasoning like subsumption (does a concept A subsume B ?), sanctioning (what are permissible roles of A ?), thesaurus lookup (what is English meaning of A ?), classification (what is the relative possition of A in the lattice of terms ?), satisfiability (is A feasible ?) and retrieval (get all instance of A and hence, all instances of concepts subsumed by A) can be automatically performed. TAMBIS domain ontology has around 1800 biological concepts and their relationships which has been created manually by a biologist and a bioinformatician over 2 years. The ontology is organized into subdomains around protein structure, function, homology, location and process. The on-

tology drives the user's experience with the system by facilitating building of only relevant queries at the concept level and mapping them into executable query plans for the sources. User does not need to know what sources were used to answer the query, providing source transparency to the user.

Queries are posed as source-independent declarative queries in terms of TaO concepts. On the other hand, the concepts supported by the local sources are available as CPL programs through Kleisli wrappers. The global concepts are mapped to the local source concepts through manually built mapping functions, which transform them into ordered (procedural) source-dependent queries for execution by the middleware. Though the maps are built manually, they have to be built only for lowest level concepts since logical reasoning can automatically build maps for the rest. The CPL program has ordered collection of function calls, and they are used by Kleisli's query reformulation subsytem to generate an executable query plan. In contrast, view integration systems like DiscoveryLink or K2/Kleisli take procedural source dependent queries directly from the user, and generate query plans from them.

TAMBIS is implemented mainly in Java. A prototype of the system has been built with 250 concepts in protein and mapped to 5 sources - SWISS-PROT for protein sequences, PROSITE for protein motifs, BLAST for protein homology, ENZYME which is a repository of enzyme classes, and CATH, a protein classifier.

6.2 e2e – An End to End Solution

e2e[1] is a comprehensive data integration solution for microarrays which allows results of microarray experiments to be evaluated for emerging patterns among groups of genes with additional insights from related analyses like pathway scoring, sequence similarity and literature text summarization to To the biologist or biological applications, *e2e* exposes a common semantic view of inter-relationship among biological concepts in the form of an XML representation called eXpressML[2]. Internally, *e2e* can use any data integration solution (like DiscoveryLink, Kleisli or natively XML-based) to retrieve data and return results corresponding to the semantic view. Currently, an *e2e* prototype allows a biologist to analyze her gene expression data in GEML or from a public site like Stanford, and discover knowledge through operations like querying on relevant annotated data represented in eXpressML using pathways data from KEGG, publication data from Medline and protein data from SWISS-PROT.

As seen in Figure 2, *e2e* envisages a two stage approach. The underlying infrastructure for *e2e* is a view integration middleware (called Enterprise In-

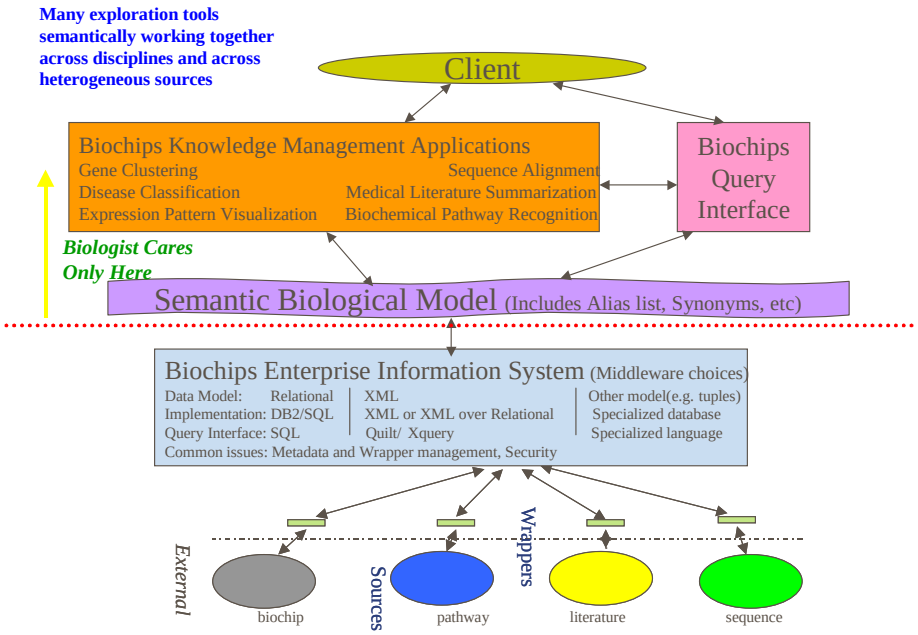

Fig. 2. *e2e* Biochips Information System Framework Architecture.

formation System to emphasize the fact that it should be able to handle large data sizes) which can retrieve either microarray experimental data or external information from publicly available biological data sources. In choosing a middleware, one has to consider the issues of uniform data model, the query language to support and availability of source wrappers. For example, with a relational data model and SQL query language, DiscoveryLink[15] is a middleware solution on commercial DB2^{TM} database while Kleisli[6] uses a complex value model of data and Collection Programming Language (CPL), but either could be used within *e2e*. In the present prototype, the data management is in XML and in-memory. eXpressML model provides the user with a common biological context to view and manipulate related data and issue XML queries in Quilt[24] through a query interface.

Additionally, *e2e* envisages an application layer where knowledge management tools are available for detecting gene expression patterns and downstream annotation of these patterns with heterogeneous information provided by the middleware. For example, some tools that can be used are: pathway visualization tools[17] for annotating gene clusters with pathway information, text summarization tools[25,16] for annotating gene clusters with biological function, and sequence alignment tools[5] for annotating gene clusters with motifs/domains. Figure 3 shows a schematic flow between KM analyses tools that a biologist may take in pursuit of discovery. *Note that the input for any KM tool is a group of genes and (optionally) eXpressML while the output is some insight about the group.* By applying diverse tools, a biologist can verify her insights with analyses spanning multidisciplinary islands of biology.

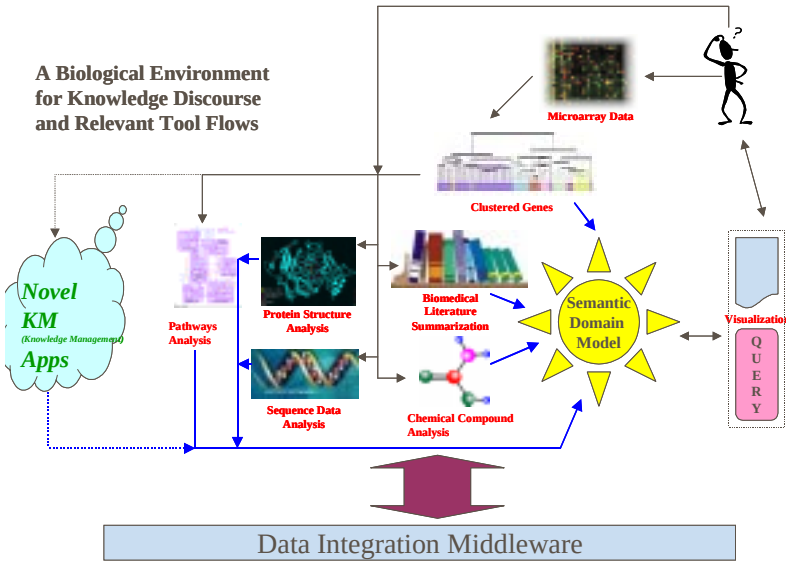

Fig. 3. A Schemata of Flow between KM Analyses in *e2e*.

7 Related Work and Conclusion

In life sciences, there are many projects on integrating heterogeneous data sources. Kleisli[8] has CPL language that allows the expression of complicated transformations across heterogeneous data sources. CPL is procedural (which makes optimization difficult) and geared toward biomedical sources. DiscoverLink [15],[12] uses SQL which is very general and well understood purpose.

Other solutions including GUS[8] and Incyte have taken a data warehousing approach to provide fast access to pre-integrated data. While warehousing approaches provide good performance, they suffer from data maintenance and replication drawbacks.

In middle level systems for information retrieval, there are many research prototypes TSIMMIS and Garlic[13]. Recently, commercial companies have emerged for content integration for multiple markets based on a XML offering. Examples are Vitria[26] and Nimble Technology [20]. Nimble has Nimble Integration Suite which uses XML-QL as the integration language. e2e also uses XML for content integration but the XML query language is QUILT.

Acknowledgments

We wish to thank the Bioinformatics group at IRL for useful discussion.

References

[1] Adak, S., Batra, V., Bhardwaj, D., Kamesam, P., Kankar, P., Kurhekar, M., and Srivastava, B. 2002. Bioinformatics for Microarrays. *Submitted for Publication.*

[2] Adak, S., Srivastava, B., Kankar, P., and Kurhekar, M. 2001. A Common Data Representation for Organizing and Managing Annotations of Biochip Expression Data. *Unpublished Technical Report.*

[3] Baker, P., Goble, C., Bechhofer, S., Paton, N., Stevens, and Brass, A. *An Ontology for Bioinformatics Applicaions.* In Bioiformatics, 15, No. 16, 510–520. 1999.

[4] Baxevanis, A. 2002. The Molecular Biology Database Collection: 2002 update. *Numcleic Acids Research, Vol. 30, No. 1.*

[5] Brazma, A., Jonassen, I., Vilo, J., and Ukkonen, E. (1998). Predicting gene regulatory elements in silico on a genomic scale. *Genome Research, 8:1202-1215.*

[6] Buneman, P., Davidson, S. Hart, K., Overton, C., and Wong, L. *A Data Transformation System for Biological Data Sources.* Proc. VLDB, pp 158–169. 1995.

[7] Cattell, R., Barry, D., Bartels, D., Berler, M., Eastman, J., Gamerman, S., Jordan, D., Springer, A., Strickland, H. and Wade, D. *The Object Database Standard: ODMG 2.0* Morgan Kaufmann Publishers. 1997.

[8] Davidson, S., Crabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, C., and Stoeckert, C. *K2/Kleisli and GUS: Experiments in Integrated Access to Genomic Data Sources.* IBM Systems Journal, March 2001. 2001.

[9] Etzold, T., and Argos, P. *SRS: An Indexing and Retrieval Tool for Flat File Data Libraries.* Computer Application of Biosciences, 9:49-57. 1993.

[10] Fujibuchi, W., Goto, S., Migimatsu, H., Uchiyama, I., Ogiwara, A., Akiyama, Y., and Kanehisa, M. *DBGET/LinkDB: an Integrated Database Retrieval System.* Pacific Sym. Biocomputing, pp 683-694. 1998.

[11] Goble, C., Stevens, R., Ng, G., Bechhofer, S., Paton, N., Baker, P., Peim, M., and Brass, A. *Transparent Access to Multiple Bioinformatics Information Sources.* IBM Systems Journal, Vol. 40, No.2, pp 532-551. 2001.

[12] IBM Almaden. *DiscoveryLink Life Science Project.* At http://www-3.ibm.com/solutions/lifesciences/discovery.html.

[13] IBM Almaden. *IBM Garlic Project.* At http://www.almaden.ibm.com/cs/garlic/homepage.html.

[14] Karp, P. 1996. A Strategy for Database Interoperation. *Journal of Computational Biology.*

[15] Haas, L., Schwarz, P., Kodali, P., Kotlar, E., Rice, J., and Swope, W. *DiscoveryLink: A system for integrated access to life sciences data sources.* IBM Systems Journal, Volume 40, Number 2, 2001. 2001.

[16] Kankar, P., Adak, S., Sarkar, A., Murari, K. and Sharma, G. (2002). MedMeSH Summarizer: Text Mining for Gene Clusters. *To appear in Proceedings of the SIAM conference in Data Mining.*

[17] Kurhekar, M., Adak, S., Jhunjhunwala, S., and Raghupathy, K. (2002). Genome-wide pathway analysis and visualization using gene expression data. *To appear in Proceedings of the Pacific Symposium of Biocomputing.*

[18] Lambrecht, E. and Kambhampati, S. *Planning for Information Gathering: A Tutorial Survey.* ASU CSE Technical Report 96-017, May 1997. 1997.

[19] Levy, A. 1998. Combining Artificial Intelligence and Databases for Data Integration. *At* *http://citeseer.nj.nec.com*

[20] Nimble. *Nimble Web Site.* At http://www.nimble.com.

[21] Object Management Group. *CORBA Specification.* At http://www.omg.org.

[22] Pottinger, R., and Levy, A. *A Scalable Algorithm for Answering Queries Using Views.* Proc. 26th VLDB, Cairo, Egypt. 2000.

[23] Rebhan, M., Chalifa-Caspi, V., Prilusky, J., and Lancet, D. *GeneCards: encyclopedia for Genes, Proteins, and Diseases.* Tech. Report, Weizmann Institute of Science, Bioinformatics Unit and Genome Center, Rehovot, Israel. 1997.

[24] Robie, D., Chamberlin, D. and Florescu, D. (2001). Quilt: an XML Query Language. *http://www.almaden.ibm.com/cs/people/chamberlin/quilt_euro.html*

[25] Shatkay, H., Edwards, S., Wilbur, J. and Bogusk, M. (2000). Genes, themes and microarrays: Using information retrieval for large-scale gene analysis. *Proceedings of ISMB'00.*

[26] Vitria. *Vitria Web Site.* At http://www.vitria.com.

[27] World Wide Web Consortium. *XML.* At http://www.w3c.org.