Logical and Semantic Database Integration

Jacob Köhler[1], Matthias Lange[2], Ralf Hofestädt[2], Steffen Schulze-Kremer[3]

1 TU Berlin, Germany, jacob.koehler@tu-berlin.de
2 Universität Magdeburg, Germany, {mlange;hofestae}@iti.cs.uni-magdeburg.de
3 Resource Center DHGP, Berlin, Germany, steffen@rzpd.de (to whom correspondence should be directed)

Abstract

Two fundamental approaches for database integration exist: the data warehouse approach attempts to physically merge data sets from several source databases, whereas database federations simultaneously query source databases online.

In this paper a database federation approach based on two components will be introduced. The MARGBench (Freier et al. 1999) is a system which, among other features, enables querying several databases in SQL by translating SQL queries into a source database specific interface. In this system SQL queries use the database field labels of the original database, i.e. fields that contain the same kind of information are differently labelled in different databases. In order to solve this problem, a second system, based on ontologies is currently being developed. This ontology system will not only include information about semantics of database fields but also contain information about databases themselves. Thus it will both facilitate database binding and intelligent database querying. The main concepts and ideas of these two systems will be explained. By using an imaginary database query it will be demonstrated how the two systems, ontology and MARGBench, will work together in order to enable querying several databases at the same time.

Introduction

In conjunction with the biotechnology boom and the human genome project (Aldhous 1990), an increasing amount of data is being generated. The amount of new data is that big that human genetics journals are increasingly reluctant to publish mutation reports (Krawczak et al. 2000). However much data is often published in publicly accessible data sources. A query to the literature database PubMed (http://www.ncbi.nlm.nih.gov/PubMed/) found 299 hits to the query: "(protein or genome) and database" of which an estimated 50% of the hits were description of distinct genome or protein databases (dated 13.05.2000). Burks 1999 gives an incomplete list of 200 WWW based data sources. Since every database uses its more or less powerful and user friendly proprietary user interface, querying these databases can be tedious. The fact that databases do not only vary in their interfaces but also in their structure and data format makes it especially difficult to combine information from several databases. Often special databases are created by merging information from other databases for the investigation of specific scientific areas, such as described by Leser et al. 1998,Agarwala et al. 2000 and Dashti et al. 1997.

An important idea of database integration is to overcome data distribution. These distributions can be intentionally or evolutionary grown. In most cases the distribution is the result of many research teams and their *ad-hoc* developed data storage methods. Ad-hoc means here the fact, that during the several research activities originated data have to be stored and the easiest, most economical method was chosen. Consequently data heterogeneity can manifest itself in several levels: a) storage and access methods b)scheme and unique identifiers and c) domain and attribute volume of the data.

Database heterogeneity can be found in storage and access methods. Database management systems, operating systems, network systems, network protocols, query languages, query interfaces and access permission differ between databases.

Databases can also vary in their data scheme and unique identifiers. Data retrieval is closely connected to scheme retrieval. Without a data scheme a structured data access is very difficult. In the worst case, databases cover gigabytes of data, without providing any scheme information. But even if the scheme information is available, identifiers, i.e. key attribute entries of databases but also the formats of equivalent entries may be inhomogeneous between databases.

Last not least, database heterogeneity can be found in the domain and attribute volume of the data. Domain in this context means semantic content of the database. Several data collections in the field of molecular biology cover many different data. These can be DNA-sequences, protein structures, enzyme regulated biochemical reaction until complex biochemical pathways and related diseases. But even databases which cover the same domains, for example enzyme databases may vary in their attributes, i.e. the properties which are stored in two database about the same enzyme is generally different. Special attention will be paid to this issue in this paper.

Consequently database integration has grown into a large area of research (Macaulay et al. 1998). In Karp 1995 four general approaches are described. These are:

- Hypertext Navigation, i.e. links between databases which can be accessed via http. In the easiest case a well organised list of links to databases is given, for example Baxevanis 2000.
- Data Warehouse, i.e. physically merging (converting, importing...) of several databases into one big database
- Multi Database Queries, i.e. querying several databases at the same time
- Federated Databases. In contrast to "Multi Database Queries", federated databases integrate database schemes in a federation layer. However, like in multi database approaches each database remains autonomous. An example of a working federation system is described by Matsuda et al. 1999.

Pro and cons of these approaches are discussed by Karp 1995. The advantages and problems of choosing the database federation approach will be compared to the datawarehouse in the discussion part of this paper. However, the aims of these approaches are the same: providing a technique to overcome the several kinds of data heterogeneity to build an unique data retrieval environment for biologists to support their research activities.

In this paper a database federation approach based on two components will be introduced. The MARGBench is a system which, among other features, enables querying several databases in SQL by translating SQL queries into a source database specific interface. The MARGBench overcomes heterogeneity issues concerning a) storage and access methods and b) scheme and unique identifiers. A prototype of the MARGBench is already working. However heterogeneity of c) domain and attribute volume of data can not yet be overcome by the MARGBench. A second ontology based component is presently being developed to overcome this kind of heterogeneity.

Ontologies for Data Integration

In Artificial Intelligence, ontologies are systems for knowledge representation based on conceptual graphs. According to Gruber an ontology is a specification of a conceptualisation (Gruber 1993b, a). One aim in AI concerning ontologies is to develop methods for knowledge representation, and how knowledge can be represented in computers. An ontology can be described as a net of nodes and edges. Nodes represent concepts, i.e. terms to be defined and edges represent relations between terms. Several systems and methods based on this idea have been developed and are reviewed by Volot et al. 1998. Several ontologies and systems for managing ontologies have been developed by now. Thus recent research focuses on interoperability and exchanging ontologies. However, many facts in the field of molecular biology is not organised in ontologies but in databases. Giudicelli and Lefranc 1999 describe a system where an ontology is being used as an intelligent front-end of a relational database which enables scientists coming from different domains to use their own terminology for database querying, thus providing an ontological view of a relational database. As mentioned before, not only semantics and the terminology used in databases are a problem, but also heterogeneity of databases. Therefore the use of ontologies for the integration of heterogeneous databases has been suggested (Kashyap and Sheth 1996, Mena et al. 1996, Schulze-Kremer 1997, Goksel and McLeod 1999).

In the subsequent paragraphs, the features needed for a database federation system based on ontologies will be summarised. In order to generate a useful and widely accepted system, the system should be created with the potential users in mind, i.e. mainly biologists. Therefore the system should provide an easy to use, yet powerful query interface which does not require in depth computer skills on the user side. Both the query interface and the database adding interface should be accessible within a web browser.

It should be possible to connect a database dynamically to the system, i.e. the database providers should be able to enter all relevant information about their database to the ontology by themselves. Database scheme information should be used if the database management system provides it. Thus it should be possible to connect the database to the system interactively with little or no manual intervention of the ontology/MARGBench operators. Many databases already have an interface, generally using http/html or SQL via JDBC (Java Database Connectivity). In order to minimise or even avoid changes having to be made in databases when they are connected to the federation system, their already existing interfaces should be used.

Even though many databases contain database fields with equivalent information, both database field labels and formats of database entries may vary. In order to enable retrieval of data from several databases, the semantics of equivalent database fields has to be defined. When thinking about operations between databases such as joining two sets of data from different databases, operations for data conversion are needed. Therefore a meta-database which stores relevant information about the databases which are accessible from the database federation system is needed. An ontology based system can be used for this. An ontology is an ideal system for the definition of the semantics of database fields and by adding extended functionality, it can also be used to organise the relevant information about the databases involved.

Ontologies tend to get complex. In order to keep them manageable, a graphical interface, representing the ontology as a web with nodes and edges is needed. It should be possible to access the ontology within a web browser so that database providers are able to enter all relevant information about their database by themselves to the system.

It seems sensitive to keep the systems as modular as possible. Both the MARGBench and the ontology component might be useful components in other contexts: ontologies may for example be used for definition of terms and the MARGBench is already being used for other purposes.

For further discussion of ontology editing (Schulze-Kremer 1998) and features for database integration by ontologies see Schulze-Kremer 1997. Another good working example for an intuitive ontology editor is given by Baker et al. 1998 and Baker et al. 1999.

Methods

MARGBench

The applied integration approach in the prototype of the MARGBench is a hybrid on the basis of the described approaches by Karp 1995.

First, the data models of the component systems, i.e. databases to be merged have to be analysed. Therefore it is necessary to analyse the local scheme's information, which is easily achieved when the component systems are based on database management systems. However, the analysis of scheme information on the basis of WWW or flat file systems can only be done manually. At present time this has to be done by the MARGBench operators. However, in the near future this can be done by the database provider by entering all relevant information to the ontology. This meets the above stated requirement to use all available scheme information.

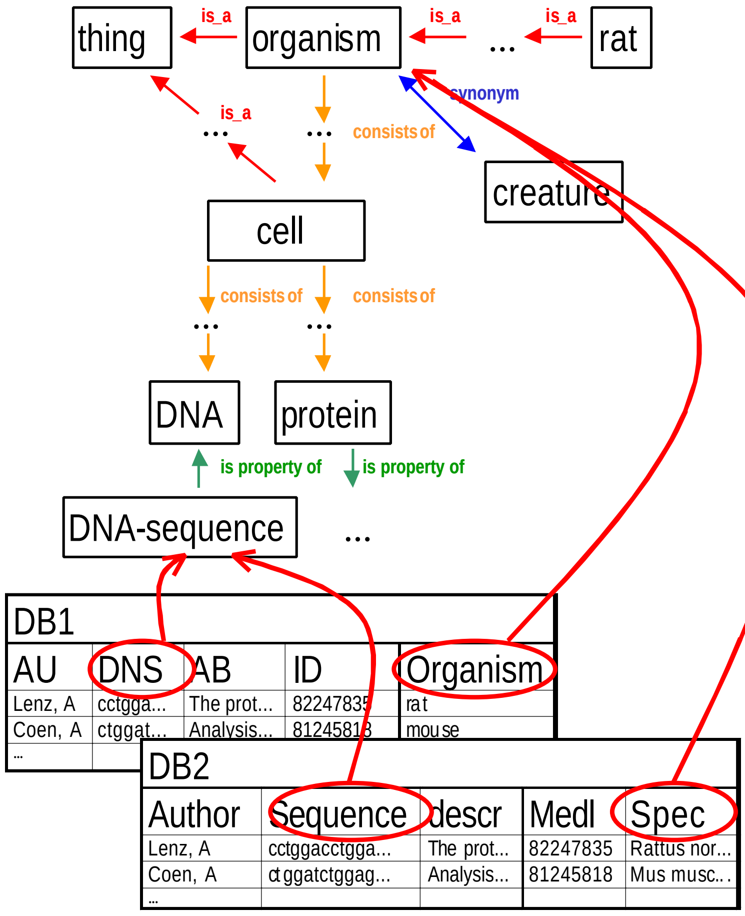
The access to the component systems is realised using special adapters, which in turn are controlled by an integration layer. Each adapter has two interfaces. One interface controls the access on the particular component system. The other transmits the queried data to the integration layer. Those adapters are implemented in a way, which allows them to access the heterogeneous systems via corresponding interfaces and query languages, i.e.

- Adapters, which access relational data bases via the Internet using JDBC
- HTML-Adapters, which analyse data from WWW pages or
- flat file Adapters, which read the data out of specific files.

These specific adapters enable the access to data sources running on heterogeneous computing systems and using different interfaces, as demanded above. Once the user submits queries to the integrated system, the data is retrieved from the component systems. Results of those queries are further submitted to the integration layer by means of the mentioned adapters. It is the integration layer which merges the results. Basis for this merger is the global scheme (see below).

Currently, adjustments of the adapters caused by scheme changes in the component systems, have to be integrated manually. Recent projects are dealing with the opportunities of an automated adapter adjustment. One possible solution for component systems based on data base management systems, could be the use of the available scheme information for a re-generation of the adapters. A second approach is the development of the ontology component which serves as a meta-database where the database provider can update the information about his database whenever it is necessary.
In the section “implementation”, a more detailed description of the integration, the architecture, and the function of the prototype, the so called *BioDataServer* is given.

Ontology
As mentioned before, the MARGBench provides SQL access via JDBC to several databases. The lobal scheme has to use the database field labels of the original database, i.e. fields that contain the same kind of information are differently labelled in different databases. In order to overcome this semantic problem, the concepts of the relevant database labels will be defined in an ontology. From these definitions, pointers will be set to all equivalent database fields. This will make it possible to localise equivalent database fields in different databases, even if the database fields are differently labelled (see figure 1).
Another functionality of the ontology system is to provide a powerful user interface to the federated databases. By making use of the is_a, synonym and homonym relations of the ontology, an “intelligent” user interface which recognises synonyms and subconcepts and checks for ambiguity of terms in a query term can be generated.



DB1

AU	DNS	AB	ID	Organism
Lenz, A	cctgga...	The prot...	82247835	rat
Coen, A	ctggat...	Analysis...	81245818	mouse
...				

DB2

Author	Sequence	descr	Medl	Spec
Lenz, A	cctggacctgga...	The prot...	82247835	Rattus nor...
Coen, A	ctggatctggag...	Analysis...	81245818	Mus musc...
...				

Figure 1: Principle of ontology based database integration. The semantics of two databases, DB1 and DB2 with equivalent content but different database attribute labels can be defined by binding them to the relevant concepts of the ontology. Concepts of the ontology are displayed in boxes, thin arrows represent relations between concepts, and thick arrows represent pointers between the ontology and the databases.

Implementation

In the subsequent paragraphs, details concerning the implementation of the MARGBench and the ontology component will be described.

MargBENCH

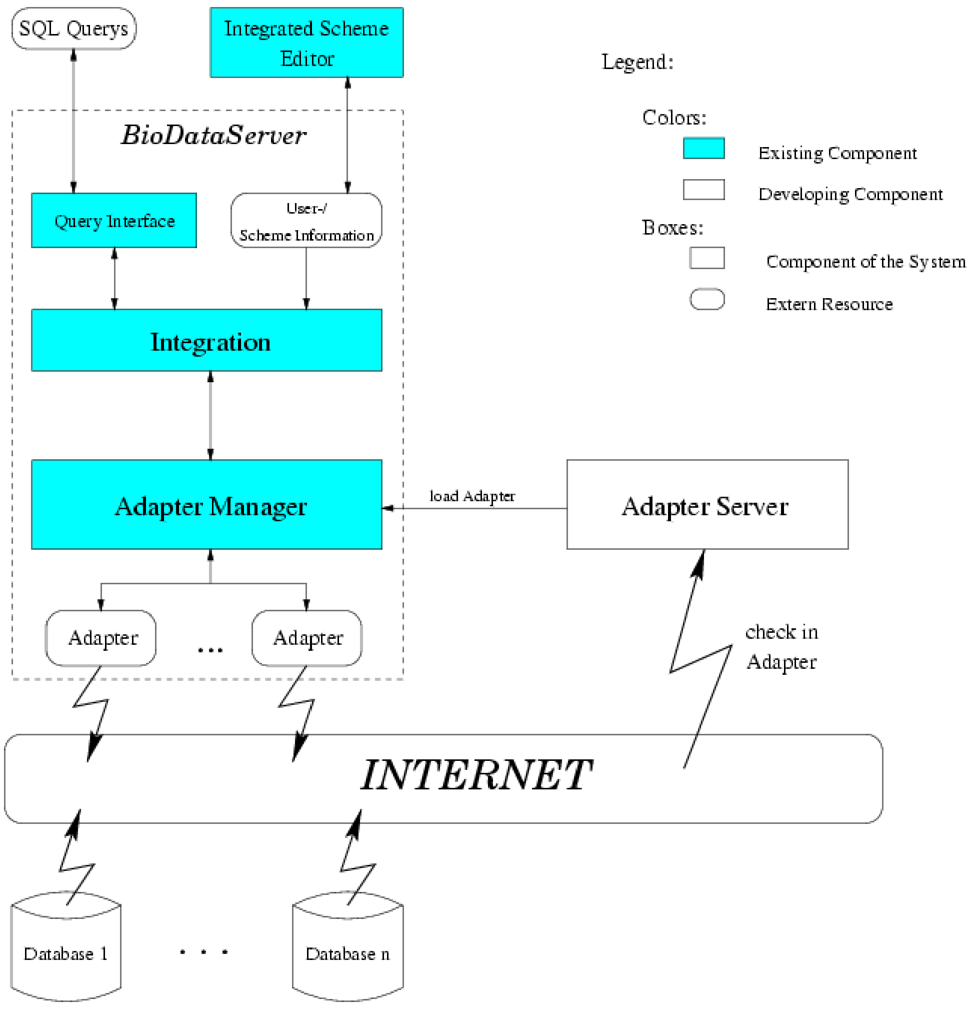


Figure 2: Integration service of the MARGBench

The integration service consists of the subsequent six components (see figure 2):

The Query Interface
By means of the query interface the user has the opportunity to interactively communicate with the *BioDataServer.* There are two distinct kinds of communication. First, a query language is provided which helps to make complex and declarative queries onto the integrate data bases. That language is a subset of SQL. Second, one can administratively control the function of the server, for instance request status information and submit commands to control the server.

User and Scheme Information
The Bio*DataServer* has been developed for multi-user operation. Respectively, an administration of several user profiles has been incorporated. This, among others, includes a specific global database scheme via selected CDB

(Component Database System). Thus the user is able to configure a integrated data scheme meeting his particular needs.

Data Integration
The core of the *BioDataServer* is the integration layer. It merges the data of the CDB logically, following the rules of the previously defined global data schemes. The required access to the CDB is transparent to the user. Therefore a query to the global data scheme is analysed and decomposed into corresponding sub queries to the CDB. Finally the resulting sub findings are merged into an integrated set of findings.

Data Base Adapter
Actual access to the data of the CDB is achieved by means of so called adapters which are essentially nothing else than drivers with a defined access interface reading the data of the CDB.

Adapter Manager
The *Adapter Manager* serves the purpose to organise the adapters of the in the system involved data bases. This includes the provision of functions to load the adapters as well as to provide a list with all currently to the system connected adapters.

Adapter Server
When planning the system's architecture a component was incorporated which administrates an archive of data base adapters. Hence the advantage of that architecture is the provision of a great variety of verified adapters. Following this line, users of molecular biological data bases is given the opportunity to publish their data bases by simply entering corresponding adapters for their data bases into the *Adapter Server.* Since a description for the functional interface of these adapters is at hand, a method exists which serves as a base for a unique access to molecular biological data bases.

Ontology component, 3 tier
Whereas the MARGBench is a database integration system which has already been implemented, the ontology component is a system which is presently being developed.

The ontology component is designed as a three tiered system:

- a relational database which stores the ontology, including the meta-database information,
- a java applet that provides as a graphical user interface from which the ontology can be edited
- and a java servlet (middle tier) connects the database system and the applet via JDBC (figure 3).

Following the model, view, controller paradigm, the relational database is the model, the servlet the controller and the applet gives a view of the ontology. This clear separation of functionality makes it possible that several users can access the ontology at the same time by giving every user his own view (applet) of the database. Thus changes to the model (relational database) can be updated in all views at runtime. The controller (servlet) performs the tasks of consistency checking, checking of access permissions, and session tracking.

The ontology component has two functions: On the one hand it serves as a meta-database for the federated database system and on the other hand it provides an intelligent user interface for querying the integrated databases. For these two tasks two different view components (applets) will have to be used.

Fundamentally, an ontology is a set of nodes and a set of edges. The edges connect the nodes, thus creating a net. An unlimited number of edge types may be defined, such as is_a, synonym, homonym, consists_of, is_database_field. This makes it easy to store an ontology as a relational database (Table 1). Note that this is a database scheme for a relational database that contains information about databases.

Node, Edge and EdgeType are all relations needed to store an ontology in a relational database system. However, in order to extend the functionality from a simple ontology to a ontology based meta-database, relations for storing information about databases are needed. The relation DB contains general information about database systems. Since each database may consist of more than one relation, information about relations will be stored in a separate relation called DBTable. DBField contains information about database attributes. The attribute DBField.PerlScript contains a Perl script or a regular expression for extraction and/or conversion of the database reply. The relation NodeFieldBinder, binds the database attributes to the relevant concepts of the ontology.

Table 1: Relational database scheme for an ontology based meta-database.

Relations	**Attributes**	**Data Type**	**Description**
Node:	NodeID		identifier for a node
	Node_Label	text	
	Description	text	
EdgeType	ETypeID		identifier for an edge type
	EtypeLabel	{is_a, synonym, homonym, ...}	
	ETypeDescription	text	
Edge:	EdgeID		identifier for an edge
	EdgeType	ETypeID	
	FromNode	NodeID	the edge connects the nodes FromNode and ToNode
	ToNode	NodeID	
DB:	DBID		identifier for a database
	DatabaseName		
	DatabaseSystem	{Oracle7, sybase, ...)	
	Interface	{SQL, http}	
	Host		only needed for SQL databases
	Port		only needed for SQL databases
	Username		needed when access to the database is restricted on the user level
	Password		
DBTable:	TableID		Database identifier
	Tablename		
	BelongsToDB	DBID	
	KeyURL		URL pointer to the key attribute of the Table, only needed for non SQL DBs
	DataURL		base URL for querying, only needed for non SQL DBs
DBField:	DBFieldID		Database Field identifier
	FieldName		
	BelongsToTable	TableID	
	PerlScript		Perlscript or regular expression for data extraction and conversion
NodeFieldBinder	FromNode	NodeID	binds database fields to the relevant nodes of the ontology
	ToDBField	DBFieldID	

However, for a “real world application”, the scheme will have to be more elaborate. It might be necessary to use two different relations for each of the DB, DBTables, DBFields relations, to make up for the differences between http and SQL databases and thus normalising the database further. Timestamps, contact address and email etc. of database provider are important information in a real world environment. In addition, mechanisms like user based access restrictions for accessing and modifying certain areas of the ontology will have to be used.

Thus it is possible to store ontologies in relational databases. However, editing or even understanding ontologies by using the user interfaces of standard database management systems, i.e. tables or forms, would be too difficult to survey. Therefore we decided to provide access to the ontology via a graphical user interface in a web browser by a java applet. In this user interface, the ontology will be displayed as a net with nodes and edges and can be edited in a user friendly way. The functionality of this ontology editor will be similar to the ontology editor which is written in PROLOG-Tcl/Tk by (Schulze-Kremer 1998). In favour of extended functionality for database management, some advanced features of the PROLOG ontology editor will not be implemented in this java version.

How can ontology and meta-database functionality be merged in a user interface? Nodes, i.e. concepts of the ontology can be expanded over an edge type. Thus a user can selectively browse a selected hierarchy, for example is_a or synonym. Even though a database field has different attributes than a node (see database scheme in Table 1), database fields will be seen like a node which can be expanded over a has_fields edge. In our relational database scheme the has_field edge is an element of the NodeField relation. When the user expands a concept over the has_field hierarchy, he will be able to access the database fields which implemented this concepts. When he selects one of these database fields, he will have access to the relevant parameters of the appertaining database.

How does database integration work, once the database provider has entered all relevant information about his database to the ontology? Since the information about the database will be stored into a relational database, this information can be retrieved using SQL, and if necessary JDBC, from any other application with access permission to the ontology meta-database. Thus the MARGBench can easily and prospectively automatically retrieve the information needed for adapter generation from this database.

How can the user query the databases? We intend to provide an intuitive yet powerful query interface which can be understood and used by non computer scientists. Fundamentally the query interface could have the subsequent structure: <Node 1>:<term 1> + <Node ...>:<term ...> + <Node n>:<term n> where <OntologyNode x> is an ontology concept and <term x> is a field entry to be searched for. Example: “Protein:amylase + Organism:mouse” means query databases for amylase in mice. When entering the query, the user can use the graphical representation of the ontology and thus browse the ontology for the concepts he is looking for. Since the user uses ontology concepts in the query terms (i.e. nodes) and not database attributes, the first step in evaluating this query term will be to retrieve the appertaining database attributes, including all subconcepts (i.e. connected by is_a edges) or synonyms (i.e. synonym edges) of organism or protein. From this list of attributes, SQL queries can be processed and used to query the MARGBench. However, before SQL queries can be sent to the MARGBench, a user scheme for the MARGBench has to be generated. The user scheme contains the global scheme for the MARGBench (see previous section). Thus each user or each application that collaborates with the MARGBench can use its own global scheme. For simultaneous querying of several databases, this user scheme would not be needed. However, for further data integration and the evaluation of inter database query terms such as joining a relation with a relation of another database, the MARGBench needs this additional information. After the user scheme has been sent to the MARGBench, the SQL queries can be sent to the MARGBench which evaluates them, i.e. translates the SQL queries into the proprietary query terms of the relevant federated database systems. The replies of the federated databases will be merged by the MARGBench and sent back to the ontology component which displays the query result to the user.

In the subsequent section an imaginary database query will be used to demonstrate how the ontology system and the MARGBench co-operate to provide the full functionality which is described in the introduction.
The above mentioned “Protein:amylase + Organism:mouse” example will be used. See also figure 3.

1.) User enters query via the ontology interface (ontology and data view).
2.) The query term is sent to the servlet component. The servlet searches the model component, i.e. the relational database for the relevant concepts by retrieving the subconcepts and synonym concepts (is_a and synonym hierarchies) via SQL queries. Homonyms will also be checked. If homonyms exist, the user will be asked to choose the correct concept. For protein the subconcept enzyme will be found. For organism the synonym concept species will be found. Depending on the size of the ontology, many more synonyms and subconcepts might be found.
3.) The servlet component retrieves from the ontology model the database fields, relations and tables information which are bound to the protein, enzyme, species and organisms concepts. This information can be accessed by querying the relations NodeFieldBinder, DBFields and DBTables from the above mentioned relational database scheme (see Table 1). It will be found that for example two databases X and Y contain relations with information about all concepts (subconcepts and synonyms) of proteins and organisms. It will be assumed that they have the relations PROTEIN{ProteinID, name, spec, description} and ENZYME{ec_number, name, org, author}. In a real query, many more databases and relations would contain information about the mentioned concepts. The number of relevant database relations might actually be so big that the user might have to be asked to select some database relations for further processing.
4.) A temporary integrated scheme for selected ontology nodes (concepts) will be generated and sent to the MARGBench. In this scheme the attribute labels to be used in the subsequent SQL query will be mapped to the labels as used in the databases. Thus it is possible to use the terminology of the ontology for the SQL query term. However each relation has to have its own name. So the relations PROTEIN will be declared as

protein1 and its attributes name and spec to protein and organism. ENZYME will be declared as protein2, its attributes name and org will be called protein and organism. In order to simplify this example, all other attributes will not be mapped.

5.) The servlet component can now create and send the appropriate SQL queries to the MARGBench: SELECT * FROM protein1 WHERE protein=amylase and organism=mouse; SELECT * FROM protein2 WHERE protein=amylase and organism=mouse
6.) The MARGBench determines which adapters will have to be used
7.) The appropriate subqueries will be generated by the subquery builder and sent to the adapters
8.) The adapters translate and send the subqueries to the appropriate databases
9.) The databases send the replies to the adapters
10.) The adapters send the replies to the "result set integration", i.e. the different formats of the replies will be unified. From databases which provide only HTML files as replies, the relevant information will be extracted.
11.) From the "Integration Layer" the MARGBench will send these replies to the servlet component of the ontology
12.) The servlet sends the reply to the user via the ontology and data view. Thus the user will get the appropriate records {ProteinID, name, spec, description} of the relation PROTEIN and the records for the attributes {ec_number, name, org, author} of the ENZYME relation.

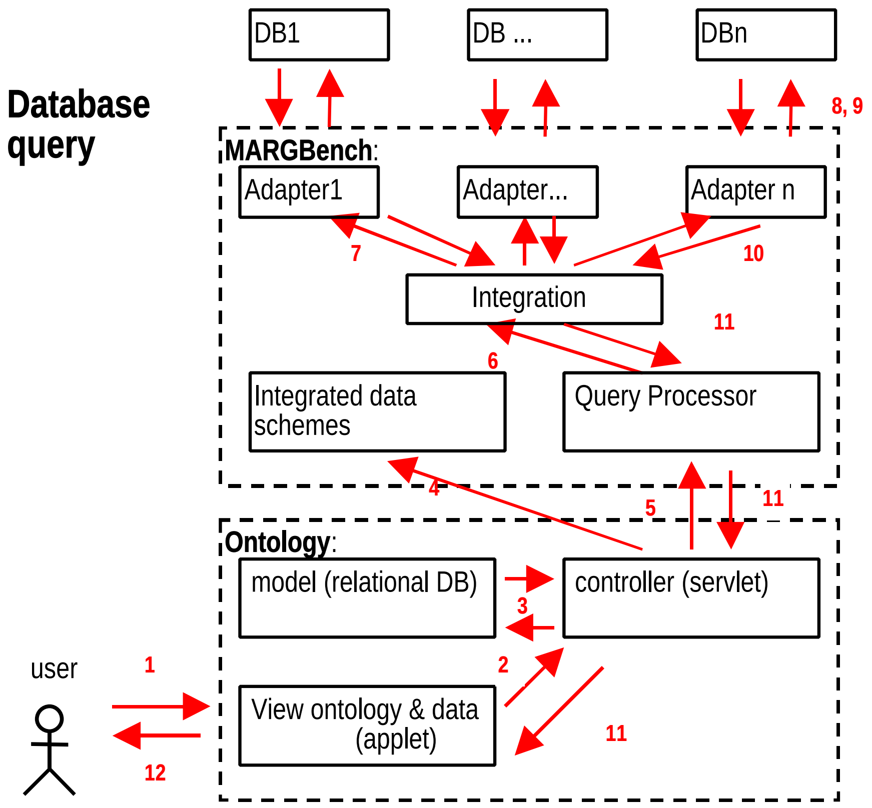


Figure 3: Components of the MARGBench and the ontology component.

Discussion

As can be seen from the example session, the combination of the MARGBench and the ontology system fulfils the system requirements as mentioned in the introduction.

However the quality of the system largely depends on the quality of the ontology. Quality can be assured by three mechanisms: Provision of a base ontology which can not be modified, access restriction and rules. A static base ontology will serve as a starting point for database providers to attach their databases to. The base ontology should contain the main concepts relevant to molecular biological science such as DNA, enzyme, cell, species etc. but also concepts important for information retrieval such as author (of a publication), abstract title etc. The base ontology makes sure that the main concepts are absolutely correct and also helps users to get an orientation in the ontology. Therefore the base ontology should only be modified when it is absolutely necessary. Since the ontology is a system that can be accessed by everybody by the internet, at least write access has to be restricted on the user level and to certain areas of the ontology. Giving everybody read access to the ontology and permission to use the query interface of the system can not lead to damage. However modifications of the ontology has to be controlled. Therefore it seems to make sense to give database providers access to the ontology in a way that they can add concepts and meta-database information. However, a database provider should only be allowed to delete or modify concepts which he himself has created. The base ontology can only be modified by MARGBench/ontology operators. Last not least, rules which at every time are valid can be given. On the one hand consistency checking of the most important relation types, such as is_a, synonym and homonym can be used to improve the quality of the ontology. Consistency checking makes sure that situations like A synonym B, B synonym C and C homonym A may not occur. This seems to be trivial, however in a large ontology these errors easily occur. Other rules like “Every node has to be connected to a concept by an is_a edge” can be given.

In the database integration approach introduced here, an easy to use yet powerful user interface was introduced and designed for a human user. However, the information in the ontology meta-database might also be useful for other applications which might want to access the ontology directly. This could be achieved for example by providing direct access to the model component of the ontology via SQL/JDBC. Special functions for finding all subconcepts and/or synonyms of an specified concept ontology could be provided.

However, not all problems concerning database heterogeneity can be solved with this system. The ontology defines the semantics of database attributes, but not terms which occur in database entries. Thus, only if the same format is used for database entries of the federated databases, “advanced” database operations as joining two records can be used. In our system, by using Perl scripts for data conversion this problem can in some cases be overcome. However, complex database entries like species names can not easily be converted. Even the systematic species nomenclature following Linné does not help in this situation: some species names are constantly being changed by phylogeneticians and even if the species names are generally identical, the spelling of the species might differ: upper case, lower case, abbreviations (sp., spec.) and subspecies make things difficult, not to mention spelling mistakes. One way to solve problems of this nature would be to use pointers to the relevant ontology concepts for definition of entries. These pointers could be URLs (Universal Resource Locators), thus entries with the same URLs are equal. The imagined URL www.molbioontology.net/~enzyme could be used to define the term enzyme. By entering this URL to a web browser the relevant concept could be displayed either graphical as a node in the ontology web or as a descriptive HTML page. This system could also be used to write text in HTML where ambiguous terms can be defined by links. However, this would require to create large knowledge repository which is permanently online. However, using this approach for homogenisation of database entries would only work between databases which use this method, i.e. many database entries would have to be updated manually with links to the appropriate ontology concepts. Another problem concerning data format are non ASCI based data fields such as image data. However, conversion of non ASCI data might be achieved by calling native conversion software on the MARGBench from the Perlscripts given by the database provider.

Many ontologies already exist. However, ontology management systems and ontology data are not always strictly separated. In some ontology systems management systems relations can not be easily modified or added since relations between concepts are often more or less hardcoded. Therefore we tried to make our ontology system as softcoded as possible, i.e. new concepts and relations can be introduced and all concepts, labels of concepts, relations and labels of relations etc. can be modified. Nonetheless, exporting hardcoded ontologies should be possible. In order to exchange ontologies between ontology management systems or maybe even to merge ontologies, a standardised exchange format is wanting. One prerequisite for ontology exchange would be to agree about a set of properties which each ontology management has to provide, such as support for one to several “root concepts”, whether or not a fixed set of new relation types has to be used, whether or not all nodes have to be connected by at least an is_a relation etc.. Once a set of ontology properties has been agreed upon, defining a unified exchange format, for example an xml “dialect” (http://www.w3.org/XML/) is a technical problem.

The database integration approach presented in this paper has several advantages but also some disadvantages when compared to the datawarehouse approach. Even though the MARGBench caches data, response of queries might in some cases be slow. This might be especially significant, when complex queries like joining two big relations of different databases have to be made. An issue which does not only concern federated databases is the loss of special i.e. proprietary database functions which require heavy computing, elaborate computing algorithms or large data datasets, such as summarised by Persson 2000. These can not be used in any integration approaches because only data and not database operations can be exported. This is due to the fact that database operations are generally closely coupled to the database management or operating system. Another topic in database integration is the quality of genome data which has recently become an issue (Harger et al. 1998, Andrade et al. 1999, Cotton and Horaitis 2000). However, the database federation approach has several advantages. It is dynamic, i.e. databases can be added to the federation system at runtime and updates of the databases take immediately effect. Modifications to the database scheme are also possible after the databases have joined the database federation, however, modifications might eventually require an update of the ontology meta-database. However, these updates can be managed by the database providers themselves. Depending on the database system and its interface, binding a new database to the federation system can be largely automated. Almost any types of database systems (object orientated, relational or file systems accessed by cgi) can join the federated system. The fact that the ontology system is not limited in contend or size, makes it possible that any number of databases independent of their contend can join the federation system. However, diskspace, network traffic and computer power limits the number of federated databases. Last not least it should be mentioned that the number of integration steps necessary for the integration of n databases is $n^2/2$, i.e. in the datawarehouse approach every database has to interface each other database whereas in the federation approach each database only has to provide one interface to the federation system.

Literature

Agarwala, R., D. L. Applegate, D. Maglott, G. D. Schuler, and A. A. Schaffer. 2000. A fast and scalable radiation hybrid map construction and integration strategy. Genome Res **10**:350-364.

Aldhous, P. 1990. Human genome project. Database goes on-line [news]. Nature **347**:9.

Andrade, M. A., N. P. Brown, C. Leroy, S. Hoersch, A. de Daruvar, C. Reich, A. Franchini, J. Tamames, A. Valencia, C. Ouzounis, and C. Sander. 1999. Automated genome sequence analysis and annotation. Bioinformatics **15**:391-412.

Baker, P. G., A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. 1998. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. An Overview. *in* sixth International Conference on Intelligent Systems for Molecular Biology, Montreal.

Baker, P. G., C. A. Goble, S. Bechhofer, N. W. Paton, R. Stevens, and A. Brass. 1999. An ontology for bioinformatics applications. Bioinformatics **15**:510-520.

Baxevanis, A. D. 2000. The molecular biology database collection: an online compilation of relevant database resources. Nucleic Acids Res **28**:1-7.

Burks, C. 1999. Molecular Biology Database List. Nucleic Acids Res **27**:1-9.

Cotton, R. G., and O. Horaitis. 2000. Quality control in the discovery, reporting, and recording of genomic variation. Hum Mutat **15**:16-21.

Dashti, A. E., S. Ghandeharizadeh, J. Stone, L. W. Swanson, and R. H. Thompson. 1997. Database challenges and solutions in neuroscientific applications. Neuroimage **5**:97-115.

Freier, A., R. Hofestdt, M. Lange, and U. Scholz. 1999. MARGBench - An Approach for Integration, Modeling and Animation of Metabolic Networks. Pages 190-194 *in* R. Giegerich, R. Hofestdt, T. Lengauer, W. Mewes, D. Schomburg, M. Vingron, and E. Wingender, editors. Proceedings of the German Conference on Bioinformatics, Hannover.

Giudicelli, V., and M. P. Lefranc. 1999. Ontology for immunogenetics: the IMGT-ONTOLOGY. Bioinformatics **15**:1047-1054.

Goksel, A., and D. McLeod. 1999. Semantic heterogeneity resolution in federated databases by metadata implantation and stepwise evolution. VLDB Journal **8**:120-132.

Gruber, T. R. 1993a. Toward principles for the design of ontologies used for knowledge sharing. *in* N. G. a. R. Poli, editor. International Workshop on Formal
Ontology. Kluwer Academic.

Gruber, T. R. 1993b. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition **5**:199-220.

Harger, C., M. Skupski, J. Bingham, A. Farmer, S. Hoisie, P. Hraber, D. Kiphart, L. Krakowski, M. McLeod, J. Schwertfeger, G. Seluja, A. Siepel, G. Singh, D. Stamper, P. Steadman, N. Thayer, R. Thompson, P. Wargo, M. Waugh, J. J. Zhuang, and P. A. Schad. 1998. The Genome Sequence DataBase (GSDB): improving data quality and data access. Nucleic Acids Res **26**:21-26.

Karp, P. D. 1995. A Strategy for Database Interoperation. J Comput Biol **2**:573-586.

Kashyap, V., and A. Sheth. 1996. Schematic and Semantic Similarities between Database Objects: A Context-based Approach. VLDB Journal **5**.

Krawczak, M., E. V. Ball, I. Fenton, P. D. Stenson, S. Abeysinghe, N. Thomas, and D. N. Cooper. 2000. Human gene mutation database-a biomedical information and research resource. Hum Mutat **15**:45-51.

Leser, U., H. Lehrach, and H. Roest Crollius. 1998. Issues in developing integrated genomic databases and application to the human X chromosome. Bioinformatics **14**:583-590.

Macaulay, J., H. Wang, and N. Goodman. 1998. A model system for studying the integration of molecular biology databases. Bioinformatics **14**:575-582.

Matsuda, H., I. Imai, M. Nakanishi, and A. Hashimoto. 1999. Querying Molecular Biology Databases by Integration Using Multiagents. IEICE TRANS. INF & SYST. **E82-D**:199-207.

Mena, E., V. Kashyap, A. Sheth, and A. Illarramendi. 1996. OBSERVER: An approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. *in* IFCIS International Conference on Cooperative Information Systems, Brussels, Belgium.

Persson, B. 2000. Bioinformatics in protein analysis [In Process Citation]. Exs **88**:215-231.

Schulze-Kremer, S. 1997. Adding semantics to genome databases: towards an ontology for molecular biology. Ismb **5**:272-275.

Schulze-Kremer, S. 1998. Ontologies for molecular biology. Pac Symp Biocomput:695-706.

Volot, F., M. Joubert, and M. Fieschi. 1998. Review of biomedical knowledge and data representation with conceptual graphs. Methods Inf Med **37**:86-96.