

Logical and Semantic Database Integration

Jacob Köhler¹, Matthias Lange², Ralf Hofestädt², Steffen Schulze-Kremer³

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

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 [1] 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 simultaneously.

Introduction

In this paper, a database federation approach based on two components will be introduced. The MARGBench is a system which, among other features, enables database integration by 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) schema 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 MARGBench. A second ontology based component is presently being developed to overcome this kind of heterogeneity.

In Artificial Intelligence, ontologies are systems for knowledge representation based on conceptual graphs. According to Gruber, an ontology is a specification of a conceptualisation [2]. 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 in [3]. Many facts in the field of molecular biology are not organised in ontologies but in databases. [4] describes 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 [5-8].

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 prospective 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 data insertion interface for ontology facts should be accessible from within a web

browser.

It should be possible to connect a database dynamically to the system by semiautomatically generated access adapters, i.e. the database providers should be able to enter all relevant information about their database to the ontology themselves. Database schema 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 modification 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 have 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 purpose. 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 become 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.

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 a further discussion of ontology editing [9] and features for database integration by ontologies see [6]. Another good working example for an intuitive ontology editor is given in [10].

Methods

MARGBench

The applied integration approach in the prototype of

the MARGBench is a hybrid on the basis of the described approaches in [11].

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 schema's information, which is easily achieved when the component systems are based on database management systems. However, the analysis of schema 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 schema 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 to the particular component system, i.e. database. The other transmits the queried data to the integration layer. The adapters are implemented in a way, which allows them to access the heterogeneous systems via corresponding interfaces and query languages, i.e. a) adapters which access relational databases via the Internet using JDBC, b) HTML-Adapters, which analyse data from WWW pages or c) flat file Adapters, which read the data out of specific files.

These specific adapters enable the homogeneous access data sources running on heterogeneous computing systems and using different interfaces. Once the user (or any application which uses the MARGBench) submits SQL 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. The integration layer merges the results based on a global schema.

Currently, adjustments of the adapters caused by schema changes in the component systems, have to be integrated manually. Recent projects are dealing with automated adapter adjustment. One possible solution could be the use of the available schema information for re-generation of the adapters.

Ontology

As mentioned before, the MARGBench provides SQL access via JDBC to several databases. The global schema of the MARGBench has to use the database field labels of the original databases, 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).

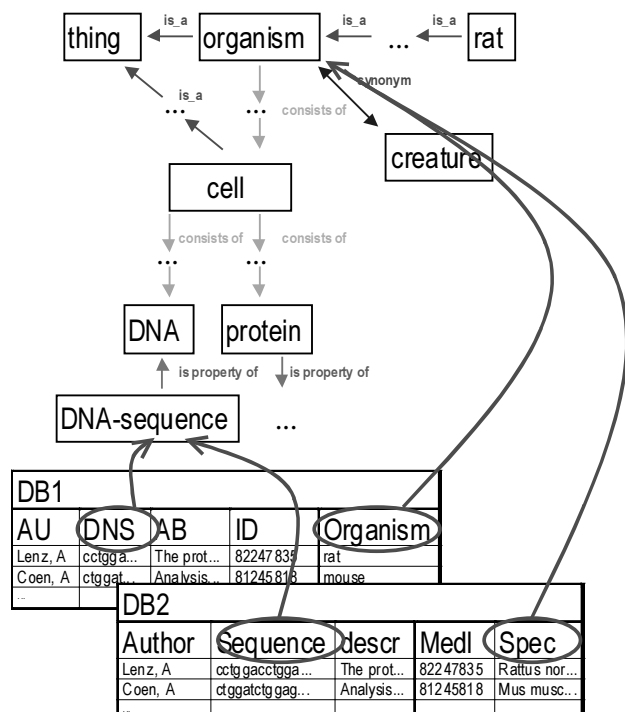


Figure 1: Principle of ontology based database integration. The semantics of two databases, DB1 and DB2 with equivalent content but different database attribute labels can be defined by binding them to the relevant concepts of the ontology. Concepts of the ontology are displayed in boxes, thin arrows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

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.

The ontology component is designed as a three tiered system: a) a relational database which stores the ontology, including the meta-database information, b) a java applet that provides a graphical user interface from which the ontology can be edited and c) a java servlet (middle tier) connects the database system and the applet via JDBC (figure 2).

Implementation

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 query term “Protein:amylase + Organism:mouse” with the meaning “query databases for amylase in mice” will be used. See also figure 2.

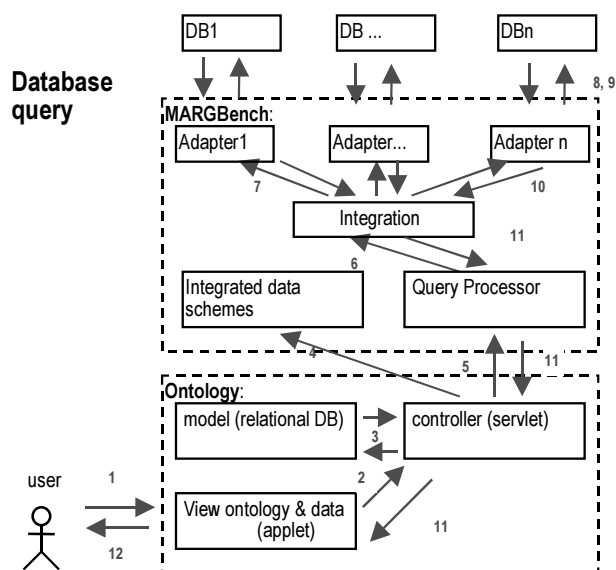


Figure 2: Components of the MARGBench and the ontology component.

- 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 the database fields, relations and tables information which are bound to the protein, enzyme, species and organisms concepts from the ontology model. This information can be accessed by querying the ontology DB component (model). 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 further 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 schema for selected ontology nodes (concepts) will be generated and sent to the MARGBench. In this schema, 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. In the integrated schema, the equivalent attributes of the relevant relations will be mapped, i.e. the integrated protein relation will be: PROTEIN_INTEGRATED (protein, organism) = {PROTEIN(name, spec); ENZYME(name, org)}. In order to simplify this example, all other attributes of PROTEIN and ENZYME will not be mapped.
- 5.) The servlet component can now create and send the appropriate SQL queries to the MARGBench: SELECT * FROM PROTEIN_INTEGRATED 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”, where the different formats of the replies will be unified. From databases which provide HTML files as replies, the relevant information will be extracted. Finally the result sets will be merged to a global result set.
- 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.

Literature

- [1] A. Freier, R. Hofestädt, M. Lange, and U. Scholz, "MARGBench - An Approach for Integration, Modeling and Animation of Metabolic Networks," presented at Proceedings of the German Conference on Bioinformatics, Hannover, 1999.
- [2] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, pp. 199-220, 1993.
- [3] F. Volot, M. Joubert, and M. Fieschi, "Review of biomedical knowledge and data representation with conceptual graphs," *Methods Inf Med*, vol. 37, pp. 86-96, 1998.
- [4] V. Giudicelli and M. P. Lefranc, "Ontology for immunogenetics: the IMGT-ONTOLOGY," *Bioinformatics*, vol. 15, pp. 1047-54, 1999.
- [5] A. Goksel and D. McLeod, "Semantic heterogeneity resolution in federated databases by metadata implantation and stepwise evolution," *VLDB Journal*, vol. 8, pp. 120-132, 1999.
- [6] S. Schulze-Kremer, "Adding semantics to genome databases: towards an ontology for molecular biology," *Ismb*, vol. 5, pp. 272-5, 1997.
- [7] V. Kashyap and A. Sheth, "Schematic and Semantic Similarities between Database Objects: A Context-based Approach," *VLDB Journal*, vol. 5, 1996.
- [8] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi, "OBSERVER: An approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies," presented at IFCIS International Conference on Cooperative Information Systems, Brussels, Belgium, 1996.
- [9] S. Schulze-Kremer, "Ontologies for molecular biology," *Pac Symp Biocomput*, pp. 695-706, 1998.
- [10] P. G. Baker, C. A. Goble, S. Bechhofer, N. W. Paton, R. Stevens, and A. Brass, "An ontology for bioinformatics applications," *Bioinformatics*, vol. 15, pp. 510-20, 1999.
- [11] P. D. Karp, "A Strategy for Database Interoperation," *J Comput Biol*, vol. 2, pp. 573-586, 1995.