Dartgrid: a Semantic Web Toolkit for Integrating Heterogeneous Relational Databases

Zhaohui Wu[1], Huajun Chen[1], Heng Wang[1], Yimin Wang[2],
Yuxin Mao[1], Jinmin Tang[1], and Cunyin Zhou[1]

[1] College of Computer Science, Zhejiang University, Hangzhou, 310027, China
{wzh,huajunsir,paulwang,maoyx,jmtang981,02rjgczcy}@zju.edu.cn
[2] Institute AIFB, University of Karlsruhe, D-76128, Germany
ywa@aifb.uni-karlsruhe.de

1 General Description

Since most of the data in big organization is stored in relational databases, for semantic web to be really useful and successful, great efforts are required to offer methods and tools to support integration of heterogeneous relational databases using semantic web technologies.

Dartgrid [3] [4] [5]. is an application development framework together with a set of practical semantic tools to facilitate the integration of heterogenous relational databases using semantic web technologies. It greatly facilitate developers (i) to interconnect distributed located legacy databases using richer semantics, (ii) to provide ontology-based query, search and navigation services as one huge distributed database, and (iii) to add additional deductive capabilities on the top to increase the usability and reusability of data.

A set of practical semantic web tools has been developed. For examples, DartMapping is a visualized mapping tool to help DBA in defining semantic mappings from heterogeneous relational schemas to RDF/OWL ontologies. DartQuery is an ontology-based query interface enabling user to specify semantic queries, and able to rewrite SPARQL semantic queries to a set of SQL queries for query rewriting. DartSearch is an ontology-based search engine enabling user to make full-text search over all databases and to navigate across the search results semantically. It is also enriched with a concept ranking mechanism to enable user to find more accurate and reliable results.

We have developed and deployed such kind of a semantic web application for China Academy of Traditional Chinese Medicine (CATCM). It semantically interconnects over 70 legacy TCM databases by a formal TCM ontology with over 70 classes and 800 properties. In this application, the TCM ontology acts as a separate semantic layer to fill up the gaps among legacy databases with heterogeneous structures. Users and machines only need to interact with the semantic layer, and the semantic interconnections allow them to start in one database, and then move around an extendable set of databases. The semantic layer also enables the system to answer semantic queries across

[3] DartGrid website: http://ccnt.zju.edu.cn/projects/dartgrid.
[4] TCM Application : http://ccnt.zju.edu.cn/projects/dartgrid/tcmgrid.html.
[5] DartGrid video demos: http://ccnt.zju.edu.cn/projects/dartgrid/document.html#Vedio

several databases such as “What diseases does this drug treat? ” or “What kind of drugs can treat this disease?”, not like the keyword-based searching mechanism provide by conventional search engines.

2 System Architecture and Technical Features

2.1 System Architecture

As Fig. 1 depicted, there are four key components in the core of DartGrid.

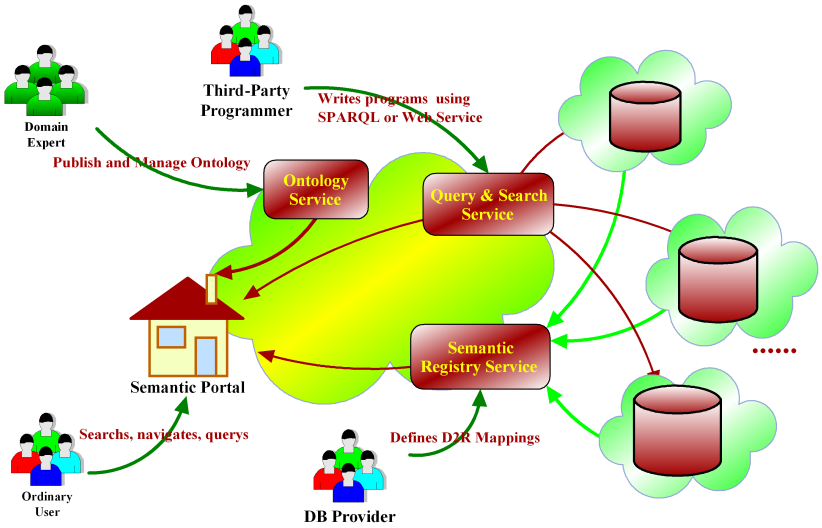

Fig. 1. System Architecture and Usage Scenarios

1. **Ontology Service** is used to expose the shared ontologies that are defined using web ontology languages. Typically, the ontology is specified by a domain expert who is also in charge of the publishing, revision, extension of the ontology.
2. **Semantic Registration Service** maintains the semantic mapping information. Typically, database providers define the mappings from relational schema to domain ontology, and submit the registration entry to this service.
3. **Semantic Query Service** is used to process SPARQL semantic queries. Firstly, it gets mapping information from semantic registration service. Next, it translates the semantic queries into a set of SQL queries and dispatch them into specific databases. Finally, the results of SQL queries will be merged and transformed back to semantically-enriched format.
4. **Search Service** supports full-text search in all databases. The search results will be statistically calculated to yield a *concepts ranking*, which help user to get more appropriate and accurate results.

2.2 Technical Features

The following features that distinguish this application from other similar systems.

1.Visualized Semantic Mapping Tool. In our system, the mappings are defined as *semantic views*, that is, each relational table is defined as a *view* over this shared ontology. Defining such kind of mappings is a labor-intensive and error-prone task. In our system, new database could be added into the system at runtime by using a visualized mapping tool, so that the system can be easily and open-ended extended. It provides many easy-of-use functionalities such as drag-and-drop mapping, mapping visualization, data source annotation and so on.

2.SPARQL Query Rewriting with Additional Inference Capabilities. A efficient query rewriting algorithm is implemented to rewrite the SPARQL queries into a set of SQL queries. This algorithm extends earlier relational and XML techniques for rewriting queries using views, with consideration of the features of web ontology languages. Besides, this algorithm is also enriched by additional inference capabilities on predicates such as *subClassOf* and *subPropertyOf*.

3.Ontology-based Semantic Query User Interface. A form-based query interface is offered to construct semantic queries over shared ontologies. It is automatically generated at runtime according to property definitions of ontology classes, and could dynamically generate a SPARQL query which will be submit to the semantic query service for query rewriting.

4.Ontology-based Search Engine with Concepts Ranking and Semantic Navigation. This Google-like search interface accepts one or more keywords and makes a complete full-text search in all databases. Users could semantically navigate in the search results, and move around an extendable set of databases based on the semantic relationships defined in the semantic layer. Meanwhile, the search system could generate a suggested list of concepts which are ranked based on their relevance to the keywords. Afterwards, users could explore into the semantic query interface of those concepts, and specify a semantic query on them to get more accurate and appropriate information.

3 Typical Steps to Build a DartGrid Application

Basically, DartGrid is an platform framework that allows users to develop their own application. Fig. 2 illustrates the directory structure of DartGrid platform. We outline the typical steps to build such an application. Reader could refer to DartGrid's developer's guide [6] for details.

1. **Build the RDF ontology.** At this step, users build the ontology for the purpose of mediating heterogenous databases.User could use any RDF/OWL-enabled ontology tool. But the recommendation is to use protege 2.1.2 which has been well tested. This step will yield an ontology file which should be copied to the *etc/onto* directory.

[6] DartGrid's Developer Guide: http://ccnt.zju.edu.cn/projects/dartgrid/dg.html

```
DartGridPlatform __src              --  query file
                 |
                  __etc__onto       --  ontology description file
                 |    |__resreg     --  resource config file
                 |    |__semreg     --  semantic mapping file
                 |
                 |__springconfig   --  applicationContext file
```

Fig. 2. Directory of DartGrid Platform

2. **Register the data sources.** At this step, users use the DartMapping tool to config the connection to a relational data source. Generally, any JDBC-enabled DBMSes are supported, but the well-tested DBMSes are Oracle and MySQL. This step will yield a resource config file which should be copied into *etc/resreg* directory. It contains all of the necessary information about physical properties of a data source such as connection string, user name and password.
3. **Define the semantic mapping from relational schema to RDF ontology.** At this step, users use the DartMapping tool to define the semantic mappings. This step will yield a semantic mapping file which should be copied into *etc/resreg* directory. It contains all of the necessary information about logical properties of a data source such as table names and their mappings to specific RDF classes.
4. **Create the semantic queries and develop your own application or user interface.** At this step, users create SPARQL semantic queries upon the ontology, develop their own application to query the distributed heterogenous databases and build the user interface to display RDF/XML-formatted query results [7].
5. **OPTIONAL: Set up the DartQuery and DartSearch User Interface.** Although users could develop their own application and user interfaces, DartGrid does offer a form-based query interface helping user to construct semantic queries and a keyword-based search interface to enable the full-text search over all databases. Readers could refer to one of DartGrid's tutorials for details. [8].

4 DartMapping: the Visualized Semantic Mapping Tools

In DartGrid, relational databases are mediated and related by a shared RDF ontology, and each relational table is mapped into one or more RDF classes. The task of defining semantic mappings from relational schema to ontologies is burdensome and erroneous.

Fig. 3 displays the visualized mapping tool we developed to facilitate the mapping task. It has five panels. The *DBRes* panel displays the relational schemas, and the *OntoSchem* panel displays the shared ontology. The *Mapping Panel* visually displays the mappings from relational schemas to ontologies. Typically, user drag tables or columns from DBRes panel, and drag classes or properties from OntoSchem panel, then drop

[7] Although DartGrid v2 does not support SPARQL, it uses a RDF-based language to describe a semantic query. For details, please refer to the DartGridv2 's developer guide.

[8] DartGrid tutorial: http://ccnt.zju.edu.cn/projects/dartgrid/dart_portal.html

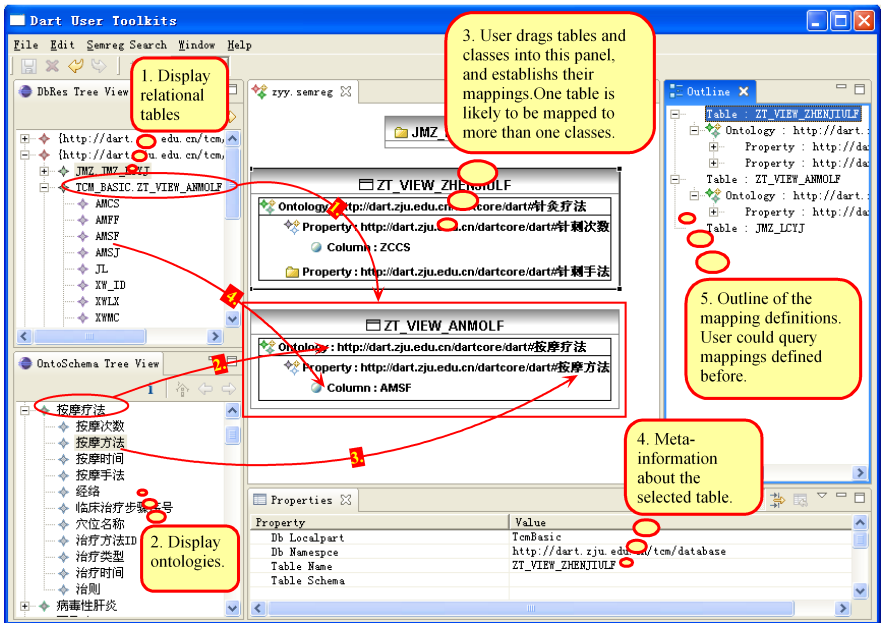

Fig. 3. Visualized Semantic Mapping Tool

them into the mapping panel to establish the mappings. By simple drag-and-drop operations, users could easily specify which classes should be mapped into a table and which property should be mapped into a table column. After these operations, the tool automatically generates a registration entry, which will then be submit to the semantic registration service. Besides, user could use the *Outline* panel to browse and query previously defined mapping information.

5 DartQuery: Ontology-based Semantic Query Interface

This form-like query interface is intended to facilitate users in constructing semantic queries such as SPARQL. The query form is automatically generated according to RDF class definitions. This design provides the extensibility of the whole system – when ontology is extended or updated , the interface could dynamically adapt to the updated shared ontology.

Fig. 4 shows the situation how a user constructs a semantic query. Starting from the *ontology view panel* on the left, user can browse the ontology tree and select the classes of interest. A query form corresponding to the property definitions of the selected class will be automatically generated and displayed in the middle. Then user can check and select the properties of interests or input query constraints into the text boxes. Accordingly, a semantic query is constructed and will be submit to the semantic query service, where the query will be rewritten into a set of SQL queries using mapping views contained in the semantic registration service.

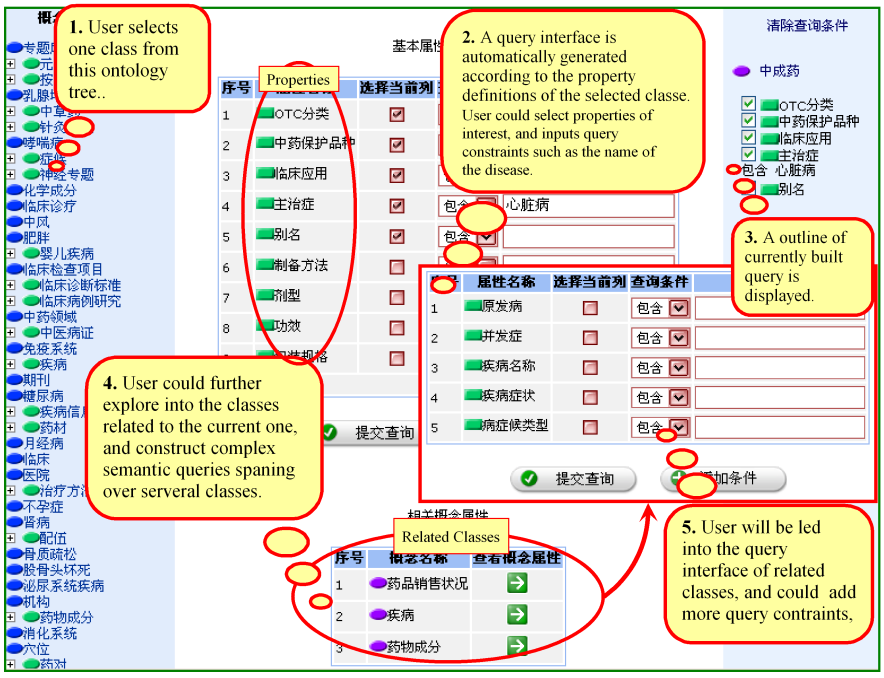

Fig. 4. Dynamic Semantic Query Portal. Please note: because many Chinese medical terminologies are only available in Chinese language and they are not always interpretable, we have annotated all the necessary parts of the figures in English.

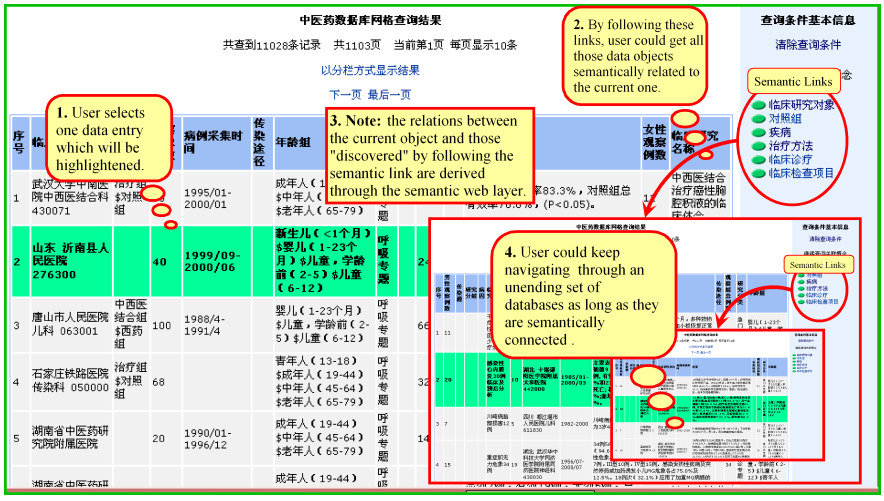

Fig. 5. Semantic navigation through the query results.

Fig. 6. Intuitive Search Portal with Concept Ranking and Semantic Navigation

User could also define more complex queries. For example, depicted in the lower-middle part of Fig. 4, user could follow the links leading to related classes of the current class, and select more properties or input new query constraints.

Fig. 5 shows the situation in which a user is navigating the query results. Starting from selecting one result highlighted, the user can find out all of the related data entries by following the semantic links. Please note that in this example, the relations between the search results and those “discovered” by following the semantic links, are derived from the semantic layer.

6 DartSearch: Ontology-based Search Interface with Concepts Ranking and Semantic Navigation

Unlike the semantic query interface, this Google-like search interface accepts one or more keywords and makes a complete full-text search in all databases. Fig. 6 shows the situation where a user performs some search operations. Starting from inputting a keyword, the user can retrieve all of those data entries containing one or more hits of that keyword. Being similar to the case of the query interface, user could also semantically navigate the search results by following the semantic links listed with each entries.

Meanwhile, the search system generates a list of suggested concepts which are displayed on the right part of the portal. They are ranked based on their relevance to the keywords. These concept links will lead the users to the semantic query interface intro-

duced in previous section. Thereafter, users could specify a semantic query to get more accurate and appropriate information.

7 Future Development

Currently, although this project is complete, several updated functionalities are still in our consideration. To be specific, we are going to enhance the mapping tools with some heuristic rules to automate the mapping task as far as possible. Otherwise, we will develop a more sophisticated mechanism to rank the data objects just like the page rank technology provided by popular search engines.