# Using RDF to query multiple SQL Databases

*Project acronym*: **QUESTION-HOW**
*Project Full Title*:**Quality Engineering Solutions via Tools, Information and Outreach for the New Highly-enriched Offerings from W3C: Evolving the Web in Europe**
*Project/Contract No.* **IST-2000-28767**

**Workpackage 1, Deliverable D1.6**

Project Manager: **Daniel Dardailler** <danield@w3.org>
Author of this document: **Michael Wilson**and **Daniel Dardailler**

**Created:** 29 August 2002. **Last updated**: 20 March 2003.

## Table of Content:

## Update

The activity for WP2 is to integrate the components of the architecture described below and developped in WP1 (and now done) and, produce the demonstration materials, ensure that the demonstration is robust, to document the demonstration to show the cost-benefits of using semantic web technologies. To provide a completed demonstration that can be used to demonstrate the benefits of the approach to potential adopters of semantic web technologies.

### Progress to date:

The plan is to finish by May 2003.

There is now a real consumer of this demonstrator in the UK Cabinet Office (Office of the e-Envoy who are W3C members) who wish to see us use it to demonstrate semantic web technologies to access their national public sector repository of XML schemas.

They are not contributing any funding to the project but we should build up good will with them as a W3C member, and promote the Semantic Web technologies.

2 small demonstrators are being produced:

- a thesaurus server in RDF (already running), and a Web Service interface. The application is designed to use RDF Query over the thesaurus server that runs as a web service to access a small set of databases (also running as web services so that we can provide an integrated answer). The thesaurus server will provide the mapping from the terms used in

the query to the terms used in the different database schema.
- a second demonstrator to access the XML schema repository and an RDF index of the Dublin Core metadata attached to each one to allow thesaurus based searching of the repository.

## Introduction

There are two directions of research for the Semantic Web: adding semantics to the Web to improve resource discovery and resource composition, and to add a layer of knowledge to the existing layers of data and information currently addressed by the Web. The second of these follows the AI tradition of encoding knowledge in rules and Knowledge Bases akin to the Expert Systems developments of the 1980's and will not be considered here as a goal in itself.

The current Web presents pages of information to users who can understand it, view the source, and cut and paste the source text and images in tools themselves to interactively compose new information (within the limits of copyright). The information is discovered by browsing links or by using search engines that index is by content. The advent of the Semantic Web provides restricted vocabularies that can be used to describe Metadata about information pages (in formats such as the Dublin Core) that can be searched by content themselves, providing a search of the information by description. Such descriptions allow search over the metadata fields such as author and date of creation which may not appear in the content of the information, and by using the restricted vocabularies, the hit rates for subject and keyword descriptions should be higher than by searching content, since there should be no or at least fewer false positives. Unfortunately, there is little constraint on information page developers to be honest in the descriptive metadata produced, and since they wish to attract as many hits as possible, they will describe pages erroneously to generate many false positives, thereby negating the advantage of metadata using restricted vocabularies. Until this trust issue can be overcome, there is no motivation for the developers of search engines to rely on the metadata since it will not improve access rates.

Data on the current web can be accessed through server side report generators which usually generate pages of information for human comsumption. Data is usually located through the same routes as information. With the advent of Web Services, data can be accessed directly through services interfaces. Since the data served through web services will itself be consumed by machines and not human users on the requesting side, there is not the same motivation to over-categorise data as there is with information. Therefore the metadata is likely to be more accurate. It is also possible to include security and trust mechanisms into the data service to avoid spam metadata. Web Services are, of course, designed not only to serve data, but also to perform services which have required inputs and specified outputs. However, these outputs are also data themselves. Web services (for data or service) can therefore more reliably be accessed through metadata and directories (e.g. UDDI) which in turn can be searched by content of interface definitions, and by descriptions in metadata. Such interface definitions and metadata can in turn be defined by restricted vocabularies themselves defined in ontologies accessible through ontology servers. Web service descriptions clearly require definitions of the service provided, the inputs and the outputs of the service, as well as quality measures on all three. Broker software can go beyond the simple identification role of information index and search software to break down search requests (for data or services) into requests for pipelines of data and services that can be composed to produce the required output from the available inputs within stated quality measures on the inputs, outputs and service itself.

So far this is the commonly held vision of the semantic web and its role in web services in 2002 in order that data and services can be located, composed and addressed by machines. This presents a realistic application of restricted vocabularies (ontologies) in order to access data and services. The evangelical message of ontologists is that restricted vocabularies in different domains can be related to one another. Therefore searches for web services or for data can be translated into the interface definition or the metadata description of one vocabulary from another. The arguments above all apply to services and to data, but to simplify the problem, from now on only data access will be considered, although the arguments still apply to services too.

Since the first development of databases, the stored data has been modelled and the data model has controlled both how the data is stored and retrieved. Data can be stored in databases of many types (relational, hierarchical, Object Oriented) and using many different domain models. Again for simplification we will only consider relational databases. Let us consider the problems of relating the models in different relational databases as a test case to demonstrate the benefits of using semantic web ontologies to define the input, output and service properties of data and web services. The problem of issuing single queries to such heterogeneous databases has long been addressed and the semantic hetewrogeneity problem here is well understood and is a clear subset of that that should be addressed by the Semantic Web technologies.

## Present State

### Heterogeneous Database Access and Semantic Schema Integration.

The data in a database are structured according to a *schema* (database definition) specified in a data definition language (DDL), and are manipulated using operations specified in a *data manipulation language* (DML). Data definition and manipulation languages are based on a *data model* that defines the semantics of the constructs and operations provided by these languages.

Managing data in multiple pre-existing databases entails dealing with their data distribution, system (e.g., DBMS) heterogeneity, and semantic (e.g., schema) heterogeneity. Approaches to managing heterogeneous databases include linking heterogeneous databases via the World Wide Web (WWW), organizing them into database federations or multidatabase systems, and constructing data warehouses. Common to these approaches is allowing component databases to preserve their *autonomy*, that is, their local definitions, applications, and policy of exchanging data with other databases (Bright et al. 1992).

Heterogeneous database systems have been traditionally classified by the type of schemas, extent of data sharing, and data access facilities they support.

Schemas supported by a heterogeneous database system include (Sheth and Larson 1990):

1. *local views* expressed representing the schemas of component databases expressed in the DDL of local databases; and
2. a *global schema* expressed in a common DDL, providing a unified view of the schemas of all component databases.

Thus, every database in a heterogeneous database system can provide a subset of its schema as its *export* schema interface to other databases; each database in turn, can have *import* schemas describing the export schemas of other databases in their local DDL (Heimbigner and McLeod 1985). The global schema of a heterogeneous database system can range from a loose collection of export schemas to a fully integrated schema. Similarly, local views of the system can range from a loose collection of import schemas to an integration of the local schema with all import schemas. For example, a *database federation* can have a global (federation) schema that provides users with a uniform view of the federation and thus insulates them from the component databases, or local views that provides users with multiple views of the federation. A *data warehouses* represents the materialization of a global schema, that is, the warehouse database defined by the global schema is loaded with data from the component databases. Unlike database federations and data warehouses, *multidatabase* systems are collections of loosely coupled databases without global schemas.

*Data sharing* in a heterogeneous database system can be at the level of:

1. linking specific data items in the component databases; or
2. generic (schema driven) correlations across component databases.

Individual data item links (e.g., hypertext links) between databases do not require or comply with schema correlations across databases. For schema correlations, data links need to be consistent with the constraints entailed by these correlations, such as inter-database referential integrity constraints.

*Data access* facilities in a heterogeneous database system can range from:

1. browsing across component databases; to
2. querying a centralized data warehouse; to
3. querying multiple databases.

Browsing across component databases is usually based on traversing WWW hyperlinks between data items in a database to data items in another database, and does not require schema correlations. Querying a data warehouse amounts to querying a single database, where the data of all component databases are represented according to the global schema of the warehouse. Querying multiple databases is carried out by expressing queries over the global schema of the heterogeneous database system or over the component database local views of the system; query translators convert queries expressed over the global schema or local views to queries for component databases. Alternatively, a heterogeneous database system can be provided with a multidatabase query language (Litwin 1987) that allows expressing queries that refer directly to elements of component databases.

A data warehouse is sometimes confused with *consolidating* heterogeneous databases into a centralized database which subsumes and replaces its component databases. However, consolidating heterogeneous databases is far more complex and expensive than constructing data warehouses. Unlike data warehouses that do not disturb component databases, consolidation eventually discards component databases and therefore requires consensus on common (global) names, data structures, values, and policy. Furthermore, all existing applications on component databases must be converted in order to comply with the new consolidated elements. This conversion process is usually very costly and not always feasible. Finally, manipulating (e.g., updating) and maintaining (e.g., reorganizing) a large database are inherently more complex processes than for smaller component databases.

## Semantic Heterogeneity

Semantic problems arise where the same term is used in different ways (e.g. bank = finance house, river side), or different terms are used for the same concept (e.g. car or auto). However, in many cases it requires complex modelling to understand the terms used in schema - e.g. does a cost include taxes or not ?

Normal DB Schema do not provide enough semantics to detect and resolve semantic heterogeneity. The usual approach is to require additional modelling information in addition to the exportable schema from a DB, and to map the set of schema onto a common high level, more general ontology.

The Semantic Web provides a mechanism for the delivery of both detailed semantic information in the form of ontologies from each organisation whose data is being accessed, and the opportunity to map terms. However the mapping of terms is usually performed by using any given term or model as a description of a space in a common ontological graph, and then using terms from the common graph as the unambiguous definitions within the heterogeneous application. In database terminology, the local ontologies would map to schema, and the global ontology would map to a global schema, and the mapping processes between ontologies would be mappings between schema undertaken by schema converters.

## Schema Converters

Global schemas supported by database federations and data warehouses are usually defined in a *common* DDL. Global schemas are more valuable if they represent a clear view of the component databases. For example, they should not be affected by implementation considerations, such as limits on the number of classes, tables, or attributes. This DBMS-independence can be achieved by defining global schemas using DDLs based on an abstract (DBMS-independent) data model, such as the semantic data models discussed in (Hull and King 1987).

Almost all heterogeneous database systems involve schema converters. Schema converters embody or are associated with data converters in that they define the way data structured according to one schema are converted to data structured according to another schema. Converting schemas and data is a non-trivial process especially when there is a substantial gap between data models.

A heterogeneous database system can involve as many converters as the number of different pairs of databases in the system, where each converter would map between a pair of specific databases. Such *database-specific* converters need to be modified whenever one of the databases changes. Alternatively, a heterogeneous database system can involve *generic* converters that do not depend on specific databases and are able to convert any database defined in a given DDL into a database defined using another DDL.

Schema and data converters can be specified using *fixed rules* or rules expressed in a *rule language*. For example, the Extended Entity-Relationship to relational DDL converter described in (Markowitz and Shoshani 1992) is based on fixed schema translation rules. The schema conversion underlying the retrofitting tool described in (Chen and Markowitz 1995c) is based on combining fixed schema translation rules with a set of schema restructuring rules. A general rule language is described in (Kosky 1995).

Finally, schema and data converters need to be qualified by *accuracy* criteria ensuring that the conversion is not losing or distorting data (Miller et al. 1993). These criteria are based on a measure of the *information capacity* preserved by the conversion.

Schema converters that lose information can lead to distorted (e.g., query) results (Miller et al. 1993). However, the information capacity criteria can in certain cases be too stringent (Kosky et al. 1995). For example, constructing global schemas sometimes involves adding information missing in (and thus *semantically enriching*) schemas of component databases or ignores information that is not relevant for constructing the global schemas. Adding missing information and/or discarding irrelevant information during schema conversion also needs to be precisely characterized.

Constructing global schemas or local views for a heterogeneous database system usually requires detecting semantic conflicts between schemas of component databases, ranging from naming conflicts and inconsistencies to detecting identical entities of interest that are represented differently. For example, homonyms can cause naming conflicts in a heterogeneous database environment, while synonyms can cause name inconsistencies. Domain conflicts can be caused by storing similar values using different units or formats in different databases. Entities of interest can be represented using various data structures in different databases, where the diversity of representations stems from different views of the data (e.g., a `Citation` can be represented only as an attribute of `Locus` or as an independent entity) and on the underlying DDL (e.g., a `Locus` can be represented as an object of a class within an object data model, but needs to be represented with one or several tables using a relational DDL). Other causes of conflicts include different ways of representing incomplete information (e.g., the meaning of nulls), and different ways of identifying objects in databases.

Resolving schema conflicts is a very complex task and may involve various methods ranging from simple renamings in order to resolve naming conflicts to schema restructurings in order to resolve structural dissimilarities. For example, in order to keep track of name correspondences, a data dictionary or thesaurus can be constructed; domain mismatches can be solved by specifying virtual attributes and domain mappings (conversions) from the real attributes to the virtual attributes (DeMichiel 1989); and structural differences can be resolved using schema (restructuring) transformations.

Alternatively, a heterogeneous database system can leave conflict resolution to users. If users are responsible for conflict resolution, a system can provide a mechanism for recording such resolutions and making them available to other users.

Correlating heterogeneous schemas can range from specifying references across database boundaries to merging schemas in order to construct a global (integrated) view (Batini et al. 1986, Buneman et al. 1992)

Inter-database references based on data (e.g., WWW hypertext) links provide a simple form of correlating databases that does not require detecting or resolving conflicts. For example, one can specify that `Maps` in a mapping database are related to `Sequences` in a sequencing database by specifying an attribute for `Maps` representing the reference to `Sequences`. However, the data representing inter-database references (e.g., *accession* numbers) must be filled in each database, for all instances that have references to instances in other databases. Inter-database references entail ensuring the validity of these references when the databases change. Such inter-database *referential integrity constraints* are hard to maintain and are seldom enforced, which may lead to inconsistent references.

Integrating heterogeneous schemas usually involves some form of conflict detection or resolution (Batini et al. 1986) as well as schema and data restructuring. For example, conflict resolution includes establishing common (integrated) names, structures, and values. Schema and data restructuring are used for eliminating structural and data redundancies. Conflict detection and resolution can be prohibitively slow if they involve too many deciding factors, or unsatisfactory to some (possibly numerous) users if it involves only one or a few deciding factors.

## Queries over heterogeneous databases

Queries in a heterogeneous database system can be expressed in the query language of the component databases or in a system-independent query language, analogous to the common DDL used for specifying global schemas.

Query interfaces are very important for facilitating the interaction of scientists with component databases. Scientists are not prepared to handle complex query languages and prefer intuitive, graphical user interfaces (GUIs). For example, most biologists are reluctant to express queries directly in SQL. It is also hard to imagine they would be willing to express queries in more powerful languages, such as CPL (Buneman et al. 1995). On the other hand, GUIs tend to support the specification of limited (e.g., form-based) queries, such as the queries supported by ACeDB.

Processing a query expressed over a global schema or a local view requires a mechanism for decomposing and translating the query into subqueries for each relevant component database, and then assembling and converting the subquery results according to the format of the global schema or local view. Such a mechanism depends on component databases providing facilities for processing external queries. Some MBDs, such as GenBank, do not provide such facilities.

Translating queries between different query languages can be very complex if the query languages are based on different data models. For example, translating queries expressed in terms of an object data model directly into relational (SQL) queries expressed in the languages supported by commercial relational DBMSs is very complex in general (Chen and Markowitz 1995b).

Query processing can involve a global query optimization stage that determines a strategy for accessing the component databases and combining the query results. Note that this global query optimization complements the query optimizers of the component databases if such optimizers are available and must take into account the peculiarities of these optimizers. The optimization techniques also depend on the type of query interface supported by the system. For example some query processing techniques are appropriate for off-line bulk data retrieval, but not for interactive queries (e.g., see Chen and Markowitz 1995b).

Retrieval queries are easier to support in a heterogeneous database environment than updates. Multidatabase updates require heterogeneous database concurrency control mechanisms that are hard to develop (Sheth and Larson 1990).

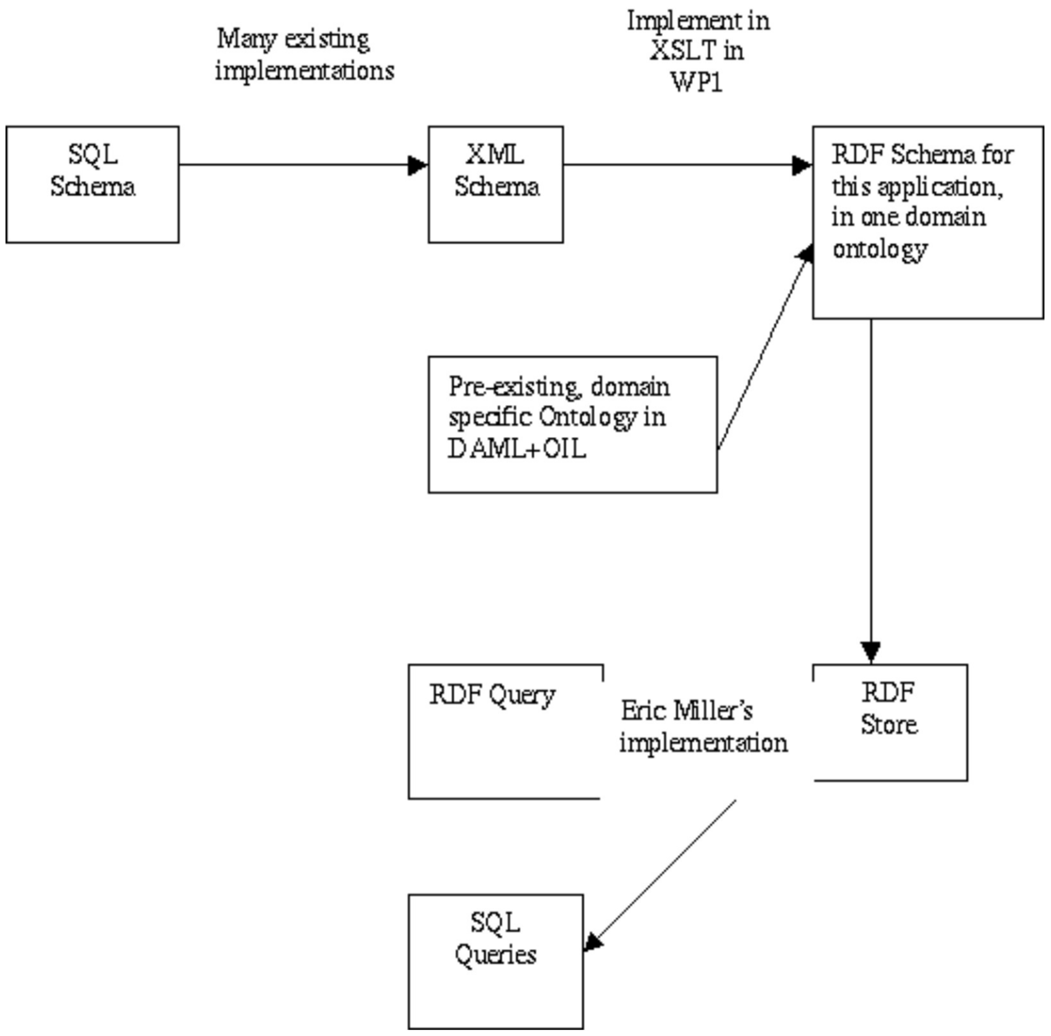
## Other Current Work

Since this workitem was proposed in 1999 several studies have been started or planned. Sheth and Meersman report the April 2002 joint NSF/EU workshop on databases and the semantic web in which most of the issues mentioned here are identified as crucial for furthering the semantic web agenda. Guha has undertaken two similar pieces of work with his small lightweitght RDF database project RDFDB and the larger on-going TAP Project at Stanford. Both of these are akin to the planned small demonstrator in this project, although they have considerably more effort involved. There is no point in the present demonstrator replicating their work so we will try to take a slightly different approach. The TAP project is addressing the use of Semantic Negotiation as a process beyond the use of tables to map between the schems elements in schemas from different heterogeneous databases as was planned for this project. The Semantic Negotiation to be used there relies on reference by description instead of reference by name, but the details of their negotiation mechanism are not yet clear - the papers the currently make available are very general and do not go into details.

## Demonstrator Study

Objective: To produce a demonstrator to show the benefits of RDF over XML Schema, raw XML, or the SQL output formats for both query and integration of information.

Scope: The demonstrator is a small integration project putting together existing open source code to provide the architecture for demonstrating the Semantic Web solution to the heterogeneous DB access problem. It will provide a small application to implement a crude description based semantic matching algorithm. However, this will not be a definitive piece of work. Rather the application will be extensible or replacable by others who devote more detailed effort to solving the problem. Similarly a small number of very small relational databases will be used. No attempt will be made to produce large databases to calculate scalability of the solution.


Many existing implementations
Implement in XSLT in WP1
SQL Schema
XML Schema
RDF Schema for this application, in one domain ontology
Pre-existing, domain specific Ontology in DAML+OIL
RDF Query
Eric Miller's implementation
RDF Store
SQL Queries

The demonstrator will provide the complete package shown above to demonstrate the benefits of using RDF as a Semantic Web solution to this problem. Many of the parts are already implemented in the DAML+Oil project, by Eric Miller at W3C (lead of W3C Semantic Web activity) or in commercially available tools (e.g. XML Spy). Other parts will require design and implementation. The whole package will need to be put together, along with example queries, DB entries etc, to produce a documented demonstration showing cost effectiveness of the approach.

The activity for WP1 is to identify the exact architecture, best existing tools at the time that can be used and to implement the missing components. The XSLT script to translate from XML Schema and pre-existing limited domain ontologies will be written. The other components will be adapted.

## Work Undertaken and Future Plans

Beyond the theoretical planning outlined above no practical work has been undertaken so far on the project since the contract was only signed just before the summer vacation and there has been no time to do any. The future plans are to implement the demonstrator in the next 6 months.

## References

- Sheth, A., and Larson, J.L. Federated databases: architectures and integration. ACM Computing Surveys, 22(3): 183--236, September 1990.
- Amit Sheth and Robert Meersman, Amicalola Report: Database and Information Systems Research Challenges and Opportunities in Semantic Web and Enterprises, Aug 2002
- Wiederhold, G. 1992. Mediators in the architecture of future information systems. IEEE Computer 25, 3 (March), 38--49.
- Express to XML schema converter
- Laks V. S. Lakshmanan, Fereidoon Sadri, Iyer N. Subramanian. SchemaSQL - A Language for Interoperability in Relational Multi-database Systems, Proc. 22nd VLDB, Mumbai, India, 1996, pp. 239-250. 12
- P. Atzeni and R. Torlone. Schema translation between heterogeneous data models in a lattice framework. In Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6), Atlanta, Georgia, 1995.
- Hull, R. 1997. Managing semantic heterogeneity in databases: A theoretical perspective. In Proc. ACM Symposium on Principles of Databases (Invited Tutorial) (1997), pp. 51--61.
- Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98), 1998.
- Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. "Mediators over Ontology-based Information Sources", 2001. (submitted for publication).
- L.M. Mackinnon, D.H. Marwick, M.H. Williams, A model for query decomposition and answer construction in heterogeneous distributed database systems, J. Intelligent Inf. Sys. 11 (1998) 69--87.
- Behrendt, W., Goble, C.A., Hutchinson, E., Jeffery K.G., Kalmus, J., Macnee, C.A., and Wilson, M.D. `Using an intelligent agent to mediate multibase information access' in Proceedings of the Special Interst Group on Cooperating Knowledge Based Systems (ed. S.M. Deen), DAKE Centre, University of Keele (1994), pp 27--43

# Deviations from plan

Was late due to signing subcontract delay. Done now.