PIONEERING TECHNOLOGIES
FOR A BETTER INTERNET

Cs3, Inc.

5777 W. Century Blvd.
Suite 1185
Los Angeles, CA 90045-5600

Phone: 310-337-3013
Fax: 310-337-3012
E-mail: info@cs3-inc.com

A Virtual Database Realization Of the Semantic Web

Dr. Donald Cohen
Dr. K. Narayanaswamy
(Contact Email: swamy@cs3-inc.com)

1. The Problem

The WWW makes a lot of data available to users but provides no mechanisms to help them process that data automatically. As a result, there is no way to combine the data from different web pages. Even in the case where the user has a conceptually simple question in mind and knows how to locate all the data, he may need to expend an inordinate amount of manual effort to retrieve and combine many pages to derive the ultimate answer to his question. This kind of work can, and should, be done automatically by software in response to a single user query.

As a simple example of this problem, consider a user viewing a list of Neil Young concert events in one page. Suppose he wishes to find the cheapest way to attend a concert from his hometown. The current WWW infrastructure requires him to copy the data from that page (dates and cities) into other web pages where he can find airfares to those cities on those dates. In fact, he may have to enter the same data into many different pages (for different airlines), along with additional constant data, such as his hometown. He then has to read the result of each query and keep track of the best combination of city, date, flight and price.

Problems such as the above were among those that were supposed to be handled by the *Semantic Web*, a compelling vision outlined in [1]. However, there is a large gap between this vision and the standards-based work described at W3C [2]. For example, in the modern Internet, decisions are rendered made based on information at various data sources. None of the standards such as RDF and SPARQL [3, 4, 5], address the kind of support required for decision makers to *react to changes* in critical data, namely triggering computations based on patterns of changes to data sources. Indeed, we believe that these languages are far too low level even for the simpler job of information retrieval for which they were created. We see a potential opportunity here for Cs3's extant Virtual Database technology called **TriggerWare** [6].

2. Technical Approach

This document provides a flavor of the overall technical approach we have in mind, while deferring major details to other documents. Our solution to the problem is to build software that allows web pages, files, etc. to be viewed as sources of data. We call the resulting system a "*Virtual Data Base*" or V*DB*. Much as relational database users can formulate queries that span different relations, users of *VDB* can formulate queries that combine the data from different web pages. Cs3's **TriggerWare** technology [6] embodies exactly this technical approach.

The *VDB* would require a small amount of human effort per data source that contains machine-readable data to identify and describe the data to be extracted. This data is then interpreted as sets of tuples of "virtual" relations[1]. Certain data sources might

[1] There is no requirement of a 1-1 correspondence between web pages and virtual relations).

offer complete database functionality, either as a web page or other interface, and should be easy to integrate into *VDB*. Therefore, we focus on the harder case, where the available data offers only a small part of full database functionality, e.g., web pages with search URL's typically offer the ability to generate tuples related to particular user inputs.

The other important aspect of *VDB* is the ability to search a space of possible algorithms in order to find the cheapest one to answer a query. The algorithms must be composed using just the interfaces that are actually available for the virtual relations involved, making this problem quite different from traditional query optimization for relational databases.

3. How VDB Can be Used for Data Integration

Cs3 personnel have been involved in developing the core of the VDB technology since the 1980s. Many of the key building blocks described above (e.g., support for non-standard implementations of relations, query optimization over such relations, triggering support for such relations) have been developed and commercialized as part of VDB by the Cs3 team. Additional technical background can be found in:

- http://www.ap5.com
- http://www.triggerware.com

We also have more technical papers about VDB that have appeared in the literature, which we can forward upon request.

In order to make data integration more convenient to accomplish through VDB we will add the following aspects to the extant VDB tools:

- **Semi-Automatic Metadata Specification:** Metadata about the nature of the data sources is to be provided by a human being. Automatic extraction of metadata is not a goal of ours, although if such tools exist, we can certainly incorporate them. We expect a small amount of human specification to describe a lot of data.
- **Query Language as Integration Mechanism:** VDB's query language, which is based on the First Order Logic, provides a natural integration mechanism. Our approach will be to enable both programmers and end users to utilize the convenience of this query language to describe how data from many different sources can be combined.
- **Support for Search and Metadata Discovery:** We do wish to make it easier for potential users to find the information of importance to them. A catalog of available data sources along with their metadata will be provided. In this regard, note that, unlike approaches like Semantic Web, we do not expect that metadata will always be provided by the owner of the data source. In our approach, metadata can be supplied by any number of 3rd parties externally to the data sources of interest.
- **Integration of External Data Sources:** In practice, we expect to support many different kinds of "adapters" for different well-known data sources to make it convenient to integrate them. We expect to offer adapters for web pages, standard relational databases, spreadsheets, and a host of other artifacts whose contents need to be integrated automatically with the data in other data sources.

4. Triggering off Changes to External Data Sources

VDB includes very sophisticated support for constraints and triggers. These kinds of rules are enforced at transactional boundaries, and make it easier to write applications using VDB We expect similar ideas to be useful for distributed, heterogeneous data sources, although it is much harder to define a transaction in such a world. We are currently investigating approaches that will support a looser form of constraints and triggers on distributed, heterogeneous data sources. We are defining the semantics of such constructs in a distributed world, and providing for different implementations for different information resources that might provide different capabilities.

5. How VDB Compares to the "Semantic Web" Work

The *Semantic Web* vision, e.g., as described in the original manifesto published in the Scientific American [1], encompasses intelligent agents that are able to access and understand nearly all data[2]. However, there is a huge gap between this grand and compelling vision and the standards based work described at W3C [2]. For one thing, it seems clear that software agents must be able to operate asynchronously, accepting and delivering relevant data as they become available or change rather than when they are requested, a mode essentially ignored so far in W3 work. Even in the more limited role of information retrieval, the standards (XML, RDF, SPARQL [3,4,5] seem unnecessarily low level.

Some of the major innovative features of our approach compared to the current RDF SPARQL approach are as follows:

- **Supporting Agents Reacting to Data Changes:** We define as "triggering" the ability to react to specific changes of interest in information resources. By building triggering facilities our research addresses an important, foundational aspect of the Semantic Web vision.
- **Use of N-Ary Relations:** We use n-ary relations and a corresponding logic-based query language, which we believe will be easier for people to use for information retrieval than one based on RDF.
- **Unification of Data and Computational Resources:** A unique feature of this research is that relations are used to model *computations* (e.g., Web services like XML-RPC and SOAP [23,24]) as well as *data*. This simplifies the language used to express queries and triggering conditions..
- **Separation of Data and Metadata:** Our approach does not require the provider of data to model and express that data in RDF. Anyone who finds data that he wants to use can provide his own model for that data. This seems particularly valuable for purposes of adoption because there is no need to wait for the owners of data to publish models of their data.
- **Queries that Involve Large Amounts of Data:** Our approach is better adapted for transmission of large amounts of data, which is sometimes necessary.

REFERENCES:

[1] *The Semantic Web*, Scientific American, May 2001, Tim Berners-Lee, James Hendler and Ora Lassila; http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2

[2] *Semantic Web*, http://www.w3.org/2001/sw/

[3] *RDF Data Access Use Cases and Requirement*, http://www.w3.org/TR/rdf-dawg-uc/

[4] *Extensible Markup Language (XML) 1.1*; http://www.w3.org/TR/2004/REC-xml11-20040204/

[5] *SPARQL Query Language for RDF*; http://www.w3.org/TR/rdf-sparql-query/

[6] TriggerWare Event Correlation Infrastructure; http://www.triggerware.com

[2] It is not clear from the literature whether the term "Semantic Web" is meant to include data other than that sent over HTTP. We take the broader view, and intend our work to apply to all information accessible by a program.