Submitted to GGF10 workshop

Design and Implementation of OGSA-WebDB
– A service-based system for making existing web databases grid-ready –

Isao Kojima and Said Mirza Pahlevi

Grid Technology Research Center(GTRC), National Institute of Advanced Industrial Science and Technology(AIST), Japan
E-mail:{kojima.mirza}@ni.aist.go.jp

Extended Abstract

1. Introduction

Distributed and heterogeneous database integration based on grid technology [1,2] is an emerging research topic, especially for scientific databases for the life sciences, or materials and geo-spatial databases. For instance, distributed data mining to explore new scientific knowledge should be possible in a grid environment that integrates many large-scale scientific databases[21].

There is currently an enormous number of databases on the web (web databases), and the problem is how to interface these existing web database resources easily in the grid environment. For example, our survey[4] found over 100 scientific web databases in our institute and over 1,000 in the area surrounding Tsukuba City, Japan. If we could interface these with a grid environment, it would be possible to accelerate research activity.

In order to achieve scientific database integration, we have developed a system called OGSA-WebDB[3]1, which integrates existing web databases in a grid environment. This extended abstract describes the design and implementation of the system.

2. OGSA-WebDB: System Architecture & Features

The purpose of OGSA-WebDB is to provide a grid service interface for existing web databases. In principle, each database query issued within the grid is converted into site-specific queries for web databases. Fig. 1 shows the architecture of the system. OGSA-WebDB consists of the following components.

Fig. 1. OGSA-WebDB Architecture

1) Proxy Database: This database delegates external web databases within the grid. Any grid application will access the proxy database instead of the original web databases. In OGSA-WebDB, we use a relational database system as the proxy database, so that each web database is represented as a relational table. Our approach is based on a virtual approach that converts an SQL query into a form query in real time; we manage the materialized proxy tables explicitly. This table can be used as a cache.

2) Mediator: The mediator bridges the gap between the proxy database and the external web databases. It accepts SQL queries (sent by a grid application) and transforms them into one or more Boolean search conditions sent to the target web databases. The mediator receives the answers from the wrappers and

---

1 We used the word “OGSA” since the system is developed on top of GT3.0(OGSI) and OGSA-DAI/OGSA-DQP.
stores them in a proxy relation. The mediator also controls parallel access and join operations over multiple web databases.

3) **Wrappers**: Separate wrappers are installed for each web database. After being invoked by the mediator, they construct and send an HTTP request to the appropriate web databases. They receive HTML documents from the web databases and convert them into a set of relational tuples to be inserted. These wrapper programs are stored using the proxy database and are managed by the database management service.

The detailed systematic procedure for processing an SQL query is shown in Fig. 2.

This architecture must deal with the following three problems.

1) **SQL Conversion to form a Query**: We assume that each web database supports Boolean conjunctions of keyword and field search conditions. Since most web databases are based on keyword searches, it is difficult to make an exact conversion from SQL to search conditions, such as comparisons. Therefore, we extract only keyword-based conditions (character matches) from the SQL statement. This is an approximate conversion that will retrieve unnecessary answers from the web. Therefore, we perform the SQL query once again on the proxy relations. This process includes a procedural overhead; however, this overhead is small, as shown below.

2) **Query Scheduling and Optimization**: An SQL query that includes join operations needs to merge datasets from several web databases. However, an SQL query can include tables without any select conditions. Some query scheduling method is needed. We assumed that the total operational cost depends on the response time of the web databases and is not a factor of how many results are returned from the web. Therefore, a simple semi-join algorithm over multiple web databases is implemented as follows:

1. Perform a conditional search in parallel.
2. For each join operation, construct a new search condition using the result of 1.
4. If all relations are queried, merge all the answers.

Figure 3 shows a snapshot of a running grid client. Note that ChemFinder requires a login before sending the query. The SQL is as follows.

```
SELECT distinct m.brand, c.formula, c.cas m.manufacture, f.strength FROM fda as f, chemfinder as c, mydruglist as m
WHERE f.name=m.generic and c.name=m.generic and f.active like "%pril%".
```

In this example, the client sends an SQL query that joins data from two web databases (Drugs@FDA[6] and ChemFinder[5]) and a local mydruglist relation.
records from ChemFinder are used for semi-joins with FDA, and the attribute values of name are converted into another search condition on the FDA web. This is performed in parallel. Finally, the answer from FDA is stored in a proxy relation and the entire SQL, including joins with a local table, is performed again.

3) Wrapper Construction: Since each web database has a different interface, development of the wrapper is an indispensable task. It is very important to provide an easy method to develop the wrapper program. OGSA-WebDB takes the following approach.

1. Combined general-purpose wrapper development system: We do not provide a specific wrapper generator tool, but interface with public/commercial wrapper generator tools, such as Republica’s XFetch Wrapper[7] and Compaq’s Web language WebL [8]. Any tool that produces a Java jar file can be integrated with the system.

2. Provide a management service: The interface with the wrapper program is also managed using proxy databases. For instance, mapping between the attributes of proxy tables and the search fields of web databases is also stored as a mapping relation. To facilitate wrapper development, this database provides a management service that is also a grid service. To add new web databases, the user enters the appropriate information and uploads the wrapper jar file. This database and the service are shared, so that users can share these wrappers and web database definitions.

Figure 4 shows a snapshot of the proxy database management tool that installs a new wrapped web database into the system. This tool also helps an administrator to install OGSA-WebDB in a grid environment by updating the grid configuration files. It is implemented as a grid client so that it can work in a shared grid environment.

3. Related Work
The web/database community has long studied data integration and data access across multiple/heterogeneous data sources[9-11]. Some methods, such as the IBM DB2 Information Integrator[12], can be used as grid/web database mediators. OGSA-WebDB has the following properties:

1) Service-based architecture: All the functions, such as SQL client and management functions, are provided with a grid service-based architecture.

2) Focus on accessing Web databases: Our architecture has a proxy database that fills the gap between SQL query language and form queries.

3) Platform independence: OGSA-WebDB can support any SQL server that OGSA-DAI supports.

4) Support of grid sign-on (GSI). GSI is mapped to database accounts and web database accounts.

4. Implementation and Performance
We have fully implemented OGSA-WebDB using Java. Currently, the system runs on top of Globus Toolkit 3[13] and OGSA-DAI 3.02[14]. Although it would be easy to implement using XML databases for query and data conversion, we adopted the OGSA-DAI relation for stability and extensibility.

Activity map file entry for the select statement is customized to invoke our mediator.
before each SQL query is processed for the proxy databases. Therefore, there is no need to modify the OGSA-DAI program code.

The proxy databases are stored in an open source MySQL[15] database. Ten sample wrappers have been written, using WebL and XFetch to provide access to several scientific, drug, patent, and bibliography web databases. In the current implementation, each database category has a separate grid service factory, so that it requires a distributed processing service (like OGSA-DQP) to integrate the database service under different factories.

![Query Processing Time](image)

Fig. 5. Query Processing Time

Figure 5 shows the query processing time for various web databases. Six web databases were used in the experiment: DBLP[16], the Collection of Computer Science Bibliographies(CSB)[17], CiteSeer[18], ChemFinder, Drugs@FDA, and Protein Data Bank (PDB)[19]. We constructed five SQL queries for each web database and determined the average query processing time. As the figure shows, the retrieval response time depends heavily on the web database, while the mediator processing time is similar for all databases. This shows that the mediator processing time is negligible given our simple query execution procedure.

5. Conclusions and Ongoing Work

This paper examined an OGSA-WebDB system that incorporates existing web databases into an OGSA-based service environment. The software was demonstrated at the SC2003 exhibition and should be released soon. In addition, some experimental query services will be provided on the internet from our website.

Current ongoing work includes:
1) Combining with OGSA-DQP[20] to support a distributed environment.
2) Devising efficient data-caching and integrity management algorithms for web databases using proxy tables.
3) Providing support for the standard web service architecture.
4) Providing an efficient database discovery method as a grid database resource information service.

References
http://www.cs.man.ac.uk/grid-db/
http://chemfinder.cambridgesoft.com/
[16] DBLP.
http://www.informatik.uni-trier.de/ley/db
http://www.gtrc.aist.go.jp/dbgrid/