Accepted at the WebNet 97 World Conference

Information Brokering on the World Wide Web

Stephane Bressan & Thomas Lee

Sloan WP# 3963 CISL WP# 97-08
June 1997

The Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02142

Abstract

Information Brokering is the process of collecting and re-distributing information. As the rich but unstructured sea of data that comprises the World Wide Web continues to grow, so too will the demand for information brokering services. This paper outlines a methodology and an architecture to support information brokering on the Web. Specifically, the two innovations which are described facilitate the construction and distribution of customized views that integrate data from a number of Web sources. An original technique is presented for the extraction of data from the Web and a solution based on JavaScript and HTML is presented to support the embedding of SQL queries within Web documents.

1.0 Introduction

In [Lev95a], the author defines **Information Brokering** as the *"business of buying and selling information as a commodity"*. Levine tracks the modern origin of information brokering to the French in 1935. The French SVP was an organization supplying information on demand over the telephone. This paper refines the definition of information brokering to mean the process of collecting and re-distributing information where **Information Brokers** are organizations which supply brokering services (Figure 1).

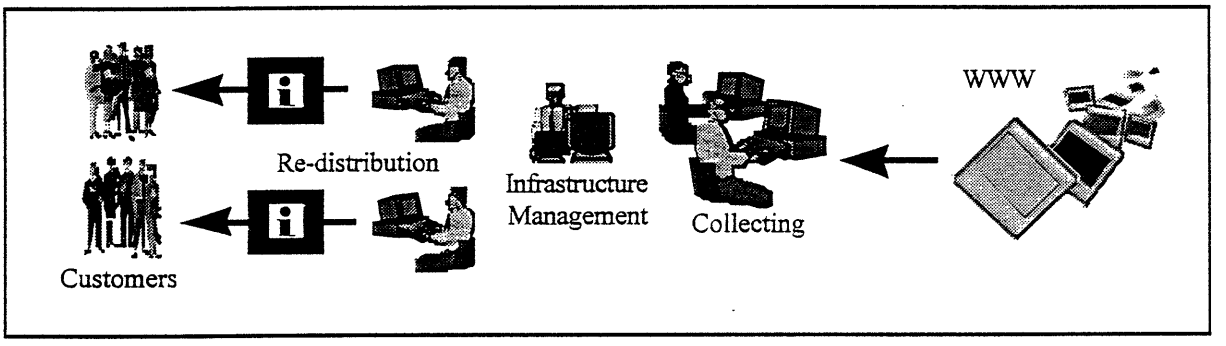

Figure 1. Information brokering

Many projects [Gar95,Lev95b,Fik96,Mar96], including the COntext INterchange (COIN) [Bre97a,Bre97b] project from which this work stems, as well as research programs like the American DARPA I3, or the European Esprit and Telematics programs, focus on the general issue of information integration. In particular, the above referenced projects and programs leverage the **Mediation** reference architecture presented in [Wie92]. Although COIN addresses the general issue of semantic integration of heterogeneous information systems, this paper focuses on the specific issue of **the collection and re-distribution of information on the World Wide Web** in Internet based public or corporate information infrastructures.

Consumers today often have specific information needs which are satisfied through the aggregation and analysis of individual data sets. While the World Wide Web offers a tremendously rich source of *data*, it fails to satisfy a user's *information needs* in at least two ways. First, information providers are constrained in their ability to flexibly present and represent data to end users. Users may be interested in graphical rather than numeric representations or aggregations rather than raw values. Providers, however, face a trade-off between flexibility and security. Existing tools sacrifice expressiveness in exchange for guarantees about more limited behavior.

Second, exacerbating the problem of representation is the challenge posed by extracting information from heterogeneous sources. Because information accessible via the Web ranges from free text to well-structured tables, users lack a uniform means for isolating and retrieving information of interest from distinct Web pages. Semi-structured sources such as HTML tables, in particular, mimic relational structures but without similar guarantees of relational behavior.

Consider, for example, the case of two financial analysts who are assessing the current state of the United States residential, long distance telephony industry. As a consequence, one analyst might like to graphically view the performance of each player in the residential, long-distance telephony market with respect to a "Communications industry index" calculated as an aggregation of share

price and shares outstanding for all market actors. A second analyst might be interested only in viewing raw figures such as closing stock price. (Figure 2)

As illustrated in the right-hand side of Figure 2, however, the data may only be available over the Web in a disaggregated form. Furthermore, the data may be indexed by company name rather than by industry sector, and may be distributed over multiple Web pages. Finally, for a specific page, users have no mechanism for identifying which values are of interest.

Information brokering is the process of identifying a user's *information needs*, *collecting* the requisite *data* from disparate sources, and then *redistributing* the formatted and processed data to the consumer. News agencies, Chambers of Commerce, and financial analysts are all examples of institutions who engage in some form of information brokering. In the example above, the analysts might perform the brokering task themselves or they might rely upon a third-party service (or internal department) to collect and compile the requisite data.

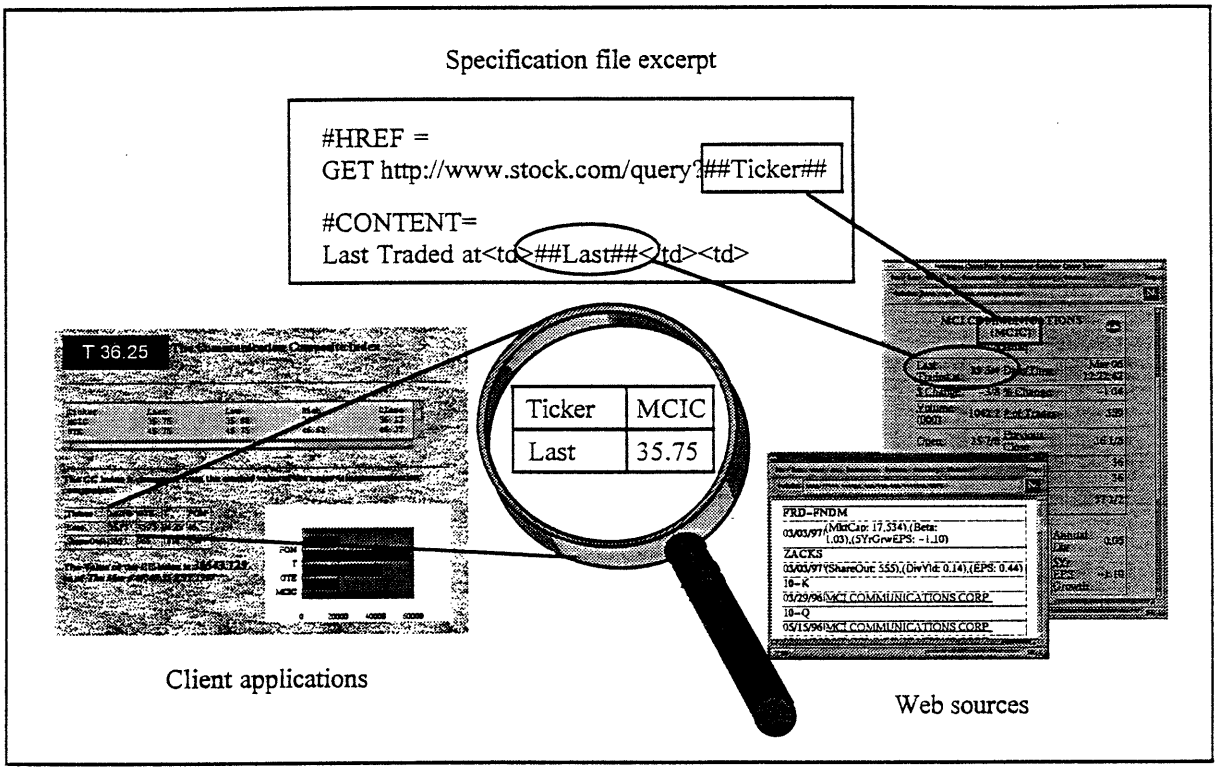

Figure 2. Example

This paper presents an architecture for Web-based information brokering. The Web brokering task is separated into three functions: collecting, redistribution, and infrastructure management. The relational data model [Ull 88] is interposed as an abstraction between the different tasks; therefore, queries are posed in a uniform manner and results may be reduced to a standard format. In section 2.0, the architecture is developed by expanding upon each of the three brokering sub-tasks. Section 3.0 discusses the motivation for this paper's approach and comments on related work and future directions. The paper concludes by speculating on the role of information brokering within the more general context of heterogeneous information integration.

Section 2.0 Broker architecture

As illustrated in Figure 3, an architecture to support information brokering may be constructed around the subdivision of the brokering process into collecting and re-distribution. Whether within a corporate intranet or over the open Internet, *Web* brokering introduces a third component, infrastructure management, that supports the collecting and redistribution sub-processes.

Section 2.1 Collecting

Web wrapping is the process of collecting or extracting data from web documents and of structuring the data into a relational form. A wrapper is a software component that serves as a gateway between the client applications and the World Wide Web. It exports relational views of some selected information on the pages, accepts SQL queries against this schema, and extracts, formats and returns the data in a relational tables.

For the purposes of this text, a document is the value returned by a Hyper Text Transfer Protocol (HTTP) request. A document is retrieved by selecting a method (usually POST or GET) and a Universal Resource Locator (URL). A URL (e.g. http://www.stock.com/query?MCIC) specifies a protocol (http), server (www.stock.com), the path on the server (/query), and a parameter string called the object body (after the question mark: MCIC). Documents corresponding to a given method and URL may be dynamically generated or may vary over time as contents are updated.

In general, the automatic extraction of data from a document is difficult. The document may not have any identifiable structure. We are here considering categories of documents which contain some observable structure such as, for instance, Hyper Text Markup Language (HTML) tables or lists that we expect to remain though some of the content varies. Data can be extracted from such documents, if the structure is known in advance, using pattern descriptions and pattern matching. Although we are currently working on techniques combining parsing of the HTML structure and regular expression pattern matching, we will only present in this paper the technique we have already implemented based on the sole pattern matching of regular expressions.

In the example of Figure 2, today's lowest price of the MCIC security is in a table cell immediately after a cell containing the string "Day Low". The regular expression pattern (in the Perl syntax) "Day Low.*</td><td>(.*)</td>" matches the sub-string "Day Low</td><td>35 5/8</td>" in the document source and binds a variable (corresponding to the sub expression in parenthesis) with the value "35 5/8". In a similar way we can match other data from the document such as the last price of the security, the highest price during the day, or the price at the previous close. A single regular expression can bind more than one variable and we can define more than one regular expression for a given document.

It is interesting to use such a description of the content of a document if we expect the document content to vary inside the limits defined by the identified structure. Such a situation is common on the Web today. However documents vary in two dimensions: over time as their content is updated, but also as the URL varies. The latter case typically corresponds to documents automatically generated by programs called via the Common Gateway Interface (cgi) for which the object body of the URL, the parameters, can change. Here, also we use the same pattern matching technique to characterize the variations in the URL. In our example, the Ticker (the identifier of the company in the stock exchange listing), MCIC, or T, or FON, or GT, are part of the object body: or "http://www.stock.com/query?(.*)".

We call a page the set of documents sharing some identifiable structure and defined by a URL pattern. A page specification contains a list of regular expressions defining the data elements in a document. The reader notices that a regular expression can find several alternative bindings on the same document.

We aim to collect the data from Web documents into a relational format. To each variable in the regular expressions, we associate an attribute of a relation. We rewrite the regular expressions using the attribute names (e.g. "Day Low.*</td><td>##Low##</td>" or "http://www.stock.com/query?ticker=##Ticker##", where ##attribute## identifies the attribute in the expression).

The relation is defined by the process of collecting data from the documents corresponding to the page specification. Alternative bindings on a document are collected in a table, the various table corresponding to multiple regular expressions are combined in a Cartesian product (we assume that they define disjoint sets attributes), and the results for each individual documents are collected in the union of all the tables.

In our example we have defined a view with the schema (Ticker, Last, Low, High, Close). One tuple in that view is (MCIC, 35 3/4, 35 5/8, 36, 36 1/8). The view contains all the corresponding tuples for each companies ticker.

We observed that the data one is interested in is often spread over multiple documents corresponding to more than one page (i.e. with various set of patterns and URL patterns). In our example, the number of outstanding shares for a given ticker can be obtained from a different document. A relation corresponds to a set of page definitions for the set of documents that one needs

to access to collect all of the data. The relation is defined as the natural join of the views for each page (i.e. the join over identically named attributes).

The number of outstanding shares and the other data collected on the first page of our example are joined on the ticker's value. If we call r1 and r2 the views corresponding to the two pages, the relation (call it security) is defined as a view:

```
DEFINE VIEW security AS SELECT r1.Ticker, r1.Last, r1.Low,
r1.High, r1.Close, r2.ShareOut WHERE r1.Ticker = r2.Ticker
```

In many practical cases, in order to keep the relationship among the different pages, we may need to use ancillary attributes (codes, html file names, values of menus in forms, etc) which are not relevant from the application point of view. For this reason we associate which each relation definition, an export schema which corresponds to the attributes visible from the application.

The definition of each relation: its name, attributes, export schema, and set of page definitions (URL pattern and regular expressions) are grouped into a unit we call the specification file.

Finally, the different specification files defining a set of relations are the parameters of the wrapper program. Given a query on the exported schemas and the specifications, the wrapper generates, optimizes, and evaluates a query execution plan. It combines the result of the query in the form of a table accompanied with additional information such as the name of the attributes, the number of answers, a time stamp on the wrapper's machine for defining the validity of the data and some other administrative data of potential use for the application.

Section 2.2 Redistribution

Redistribution involves posing queries, integrating the data retrieved from one or more Web wrappers, and formatting the data to meet a client application's requirements. Redistribution may also require additional data processing. In the earlier example, the financial analyst calculating the "Communications industry index," stock values from all residential, long-distance market actors are aggregated. The solution introduced here leverages "server side includes" (SSI) and JavaScript to submit queries and to process the results.

Queries are embedded within Web documents and submitted via SSI. A single HTML command both defines the query and declares a handle on the result. For the telecommunications industry analysis referenced above, the query might appear as:

```
<!-- #exec cmd="wrapper query=SELECT Ticker, ShareOut, Last FROM security
WHERE Ticker in (T, MCIC, FON, GTE); handle=tel_index-->
```

When a client application requests the document containing a query, the Web server invokes the query and returns to the client a document where each command line is replaced by the query result.

Rather than returning query results as HTML formatted text which is directly substituted into a Web document, the SSI introduced in this paper returns a JavaScript program. The JavaScript program defines a JavaScript object which is referenced by the handle in the SSI declaration. The result object contains a query result table, attribute names, data types, and administrative information provided by the wrapper such as time-stamps. Values are accessed and formatted by applying any number of primitive or advanced JavaScript functions to the JavaScript result object. Basic functions are provided as direct methods of the result object, and advanced functions could be defined by Web page designers or loaded via SSI from libraries.

As illustrated in the left-hand side of Figure 2, the combination of SSI and JavaScript demonstrated in this paper offers tremendous flexibility with respect to both data presentation and data re-use. Values may be aggregated or displayed in raw form. Data may be formatted in tables or graphs. The Communications Composite Index in Figure 2 is generated by the following HTML-embedded JavaScript program:

```
<H3> The Communications Composite Index is
<BLINK><SCRIPT>
index=0;
for (i=1; i < size(tel_index); i++){
  index = tel_index[i][2] * tel_index[i][3];}
document.writeln(index);
</SCRIPT></BLINK>
<SCRIPT>timeStamp(tel_index);</SCRIPT>
</H3>
```

Data from a single result object may be reused in multiple portions of a document without re-submitting the same query or a sub-query (SSI). For example, the bar chart on the left-hand side of Figure 2 was generated by passing the result-object to a public domain Java applet. This combination of SSI and JavaScript offers maximum flexibility in presentation and ex-post data manipulation while minimizing the number of external calls and leaving much of the formatting to the client application (JavaScript).

Section 2.3 Infrastructure management

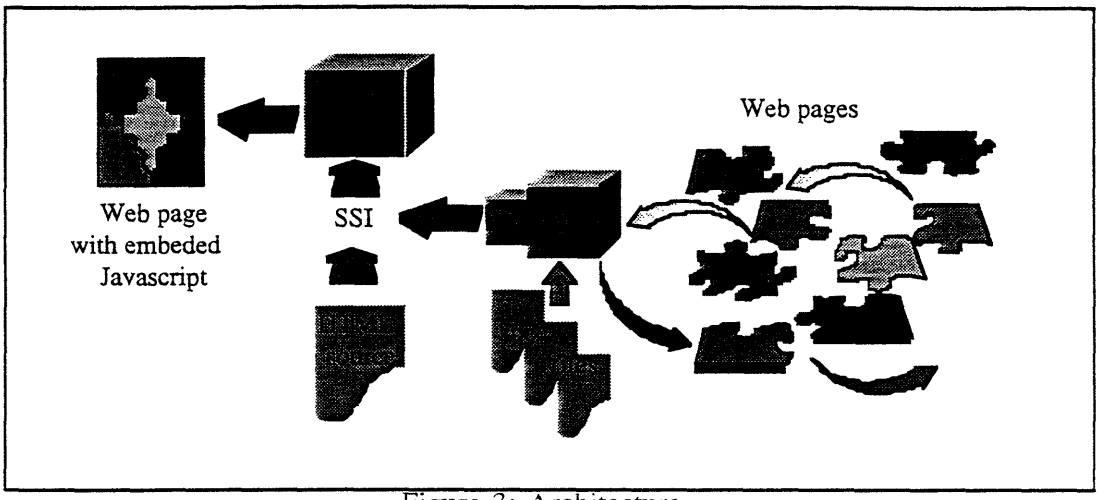

Figure 3: Architecture.

Figure 3 summarizes the architecture we propose. The structure of the information located on various disparate Web pages is described in the specification files. The wrappers receiving a query extract, combine, and as required by the query and the relational schema. Queries are inserted as SSI-commands in the HTML source. The commands are processed the wrappers on request of the Web server. The result of the SSI-processing is a JavaScript object available on the document served. The data are displayed in different formats and combinations by means of JavaScript functions embedded in the HTML pages and processed on the client machine. Different designers can freely combine the data available in appropriate formats on their Web pages.

Section 3.0 Discussion and conclusion

3.1 Related work

Wrapping
There currently exist a number of broader information integration efforts that employ some form of non-intrusive wrapper for exporting a uniform query interface to and extracting data from disparate sources. Wrapper technologies differ with respect to the query language interface, the kinds of sources supported, and the functionality delegated to wrappers.

InfoSleuth agents [Woe96] accept KQML queries and access any number of source interfaces including the Web and traditional relational database interfaces. TSIMMIS [Gar95] wrappers process Mediator Specification Language queries on OEM objects. TSIMMIS wrappers, which map sources into OEM objects and create optimized query execution plans, have been demonstrated for the LORE database and network information services like finger and whois. SIMS leverages a LOOM knowledge representation to wrap Oracle databases and LOOM knowledge bases by mapping data sources to subclasses of a LOOM ontology. Though queries are posed as LOOM statements, the wrapper itself does little processing, relying upon other components of the SIMS system to plan and optimize.

Redistribution
Redistribution is a particularly challenging problem for information brokers because it is difficult to determine a priori the dimensions of a consumer's demands. Even within a single institution, as in

the case of the two telecommunications financial analysts above or across a corporate, departments and divisions may present different aggregation and formatting requirements. Conventional solutions rely either upon CGI scripts or SSI.

CGI solutions are overloaded so that scripts not only request and retrieve data but also format the resulting data within a Web document. Such an architecture is a drawback when the same data is re-used in more than one application. The proliferation of scripts complicates quality control and compounds the information broker's security vulnerabilities due to multiple, repeated gateways between the outside world and the broker's servers.

SSI solutions also typically overload a single operation with query request, retrieval, and format. SSI calls typically return HTML to facilitate direct substitution into the encompassing HTML. Moreover, as alluded to above, SSIs can introduce querying inefficiencies. Rather than submitting a single query to an external source and then caching the results (JavaScript object or CGI variables), multiple SSI invocations are required. Even re-using the same data requires a new call. Finally, SSIs alone support only limited formatting capabilities. Standard extensions call for passing formatting parameters in the SSI ultimately reducing the call to a general scripting language.

3.2 Limitations and future work

As detailed, the current information broker is limited in three dimensions. First, the absence of a means to express knowledge about the page's content and the lack of such information limits the opportunities for optimizing the process of data collection by selecting the most appropriate documents.

Second, although the combination of SSI and JavaScript affords great flexibility in data formatting and presentation, client applications are ultimately limited to rendering the JavaScripts and HTML returned to them by information brokers. A logical extension is to therefore enable clients to dynamically reformat values.

A related limitation and attendant extension is the ability for end users to parameterize queries. In the current architecture, because queries are invoked from the information broker's Web server through an SSI, queries are articulated in advance and maintained as static pages by the broker. The third dimension of future work calls for enabling client applications to dynamically structure and submit queries through the information broker directly to wrappers, using a gateway (a Java applet) monitored from the client and collaborating with client-side JavaScripts.

The described architecture has been implemented and is currently deployed within the Context Interchange system.

References

[Bre97a] Bressan,S., Fynn, K., Goh, C., Madnick, S., Pena, T., and Siegel, M. *"Overview of a Prolog Implementation of the Context Interchange Mediator"*. Proc. of the Intl. Conf. on Practical Applications of Prolog. 1997.

[Bre97b] Bressan, S. and al. "The Context Interchange Mediator Prototype". Proc of SIGMOD97. 1997.

[Fik96] Fikes, R. *"Network based Information Brokering"*. http://www--ksl.stanford.edu/kst/info-broker.html. 1996.

[Gar95] Garcia-Molina, H. *"The TSIMMIS Approach to Mediation: Data Models and Languages"*. Proc. of the Conf. on Next Generation Information Technologies and Systems. 1995.

[Lev95a] Levine, M. *"A Brief History of Information Brokering"*. American Society for Information System Bulletin. February 1995.

[Lev95b] Levy, A., Srivastava, D., and Kirk, T. *"Data Model and Query Evaluation in Global Information Systems"*. J. of Intelligent Information Systems. 1995.

[Mar96] Martin, D. *"The Information Broker Project"*. http://www.ai.sri.com/~martin/broker/techrep/memo.ps.gz.1996.

[Sch96] Schum, A. "Open Database Connectivity of the Context Interchange System". MIT Master thesis. Dpt. of Elect. Eng. 1996.

[Tom95] Tomasic, A., Rashid, L., and Valduriez, P. *"Scaling Heterogeneous databases and the Design of DISCO"*. Proc. of the Intl. Conf. on Distributed Computing Systems. 1995.

[Ull88] Ullman, J. *Principles of Database and Knowledge-base Systems, Volume 1*, Computer Science Press, Rockville, MD, 1988.

[Wie92] Wiederhold, G. *"Mediation in the Architecture of Future Information Systems"*. Computer, 23(3). 1992.

[Woe96] Woelk, D. *"Infosleuth"* . http://www.mcc.com/projects/infosleuth 1996.