Web services and data integration

Serge Abiteboul
INRIA

Omar Benjelloun
INRIA

Tova Milo
Tel Aviv U. and INRIA

July 26, 2002

Abstract

The developments of XML [4] and web services [3] are changing radically the art of data integration. We briefly describe web services and consider some of their impact on data integration. We describe some work going on at INRIA on Active XML, that is XML extended by allowing the embedding of calls to web services. We argue that XML and web services provide the proper infrastructure for data integration at the web scale.

1 Introduction

Data integration has been a very hot topic both for research and industry for a long while. In this paper, we argue that XML and web services provide the proper infrastructure for data integration at the web scale. First XML provides a nice mediation model, i.e., a lingua franqua, or more precisely a syntax, that most pieces of software can (or will soon) understand. Second, web services provide the adequate abstraction level to describe the various actors in data integration such as wrappers or mediators and the communications between them. We illustrate this thesis with an on-going software project at INRIA, namely Active XML, that is based on the embedding of web service calls inside XML documents.

What is the information that may be found on the web and is candidate for integration:

- HTML still forms the backbone of the web, but popular document formats such as DOC, PDF, BS are massively present.
- A new web exchange format, namely XML [4], promoted by the w3c [9] and the industry, is radically changing the web. XML marries, in

some sense, the document world with the database world [1]. XML can be used to represent documents, but also structured or semistructured data [1].

- A lot of web data is typically retrieved as a result of queries, in particular queries to database systems. Typically, this means that data (query parameters) has to be fed to obtain more data. The term deep web or hidden web are used for such accesses typically via forms, scripts or using a new standard for web services, SOAP [3].
- Finally, multimedia formats for images, sound, video are crucial components of the web that we will not discuss here.

In this paper, we do not consider document formats except for XML, and the problem of querying multimedia information. We will assume that appropriate wrappers have been developed to transform such information in XML. On the other hand, we are deeply concerned with access to database or in general to information services. We will assume that such services are also wrapped inside SOAP web services. More precisely, we will assume that some web services provide the necessary translation. Thus, although we ignore here some critical issue, namely how this wrapping is performed, we can now live in a more unified world with XML as the unique data model and web services as the distributed model of computation. As we shall see, this provides a uniform infrastructure for data integration based on the two main standards:

1. A standard of data exchange, XML, a mark-up language for tree-structured data.
2. A standards for web services, namely SOAP. (Or more precisely, as we will see a family of standards.)

We forcus here on an infrastructure for data integration. We do not consider the internal logic of the wrappers. Similarly, we are not concerned with the logic of integration, e.g., is it in a wraehousing or mediating approach, it is based on local-as-view or global-as-view. These important aspect will guide the selection of appropriate web services that will be used by our system.

The paper is organized as follows. We assume some knowledge of XML since this is now a well-accepted standard. We briefly present web services. We then consider how data integration can benefit from XML and web services. Finally, we mention a particular research project at INRIA, namely Active XML, that is based on embedding web service calls inside Active XML documents.

2 Web services in short

Web Services (successors of Corba and DCOM) are the next step in the evolution of the web. They allow active objects to be placed on web sites providing distributed services to potential clients. Although most of the noise comes from e-commerce, one of their main current uses is for the management of distributed information. Distributed database systems always suffered from platform and software incompatibilities. Web services are not inventing anything new, but they are bringing an important breakthrough to distributed data management simply because they propose web solutions that can be easily deployed independently of the nature of the machine, the operating system and the application languages.

In short, the general setting benefits from a variety of standards around web services: setting:

Find the service/information First one can query some yellow-pages to find the provider of some information. This is achieved for instance with a standard, UDDI [8], Universal Discovery Description and Integration. UDDI is a specification for distributed web-based information registries of web Services. UDDI is also a publicly accessible set of implementations.

Understand how to obtain it This is achieved in particular using WSDL [6], the web Service Definition Language, a language for describing web service interfaces, something like Corba's IDL for the web. WSDL is an XML format for describing network services based on operations and messages. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define a functionality of a service.

Get and use it We can then directly obtain the information or more interestingly use it in our (mediator or warehouse) system using SOAP, the Simple Object Access Protocol. SOAP is an XML based lightweight protocol for exchange of information in a distributed environment. In particular, it allows to specify the (XML) types of arguments and service results. SOAP can be used, in particular, in combination with HTTP.

Of course life is more complicated, so one will probably have to sequence operations (see Web Service Flow Language) and consider issues such as privacy policy, cost, etc. (see Web Service Endpoint Language).

3 Data integration

When we consider the activity of data integration, we find ingredients that are similar to those of web services:

Find the data sources of interest in some directories;

Find the descriptions of the sources;

If needed, find wrappers for these sources;

Find a mediator/warehouse or construct one – this is the core of the technology of the data integration technology;

Run a query.

All these pieces of software may run on different machines and may be viewed as web services. Note that we may assume that most of these pieces of software speak XML with the exception of the sources that typically translate from some internal format to XML. Observe also that some wrappers may translate from XML to XML, e.g., from one local XML dialect for chemical products to a standard XML dialect for such data, i.e., the Chemical Markup Language.

Consider Figure 1. Form a logical viewpoint, the mediator integrates data coming from 3 sources. First, source1 exports some XML and the mediator can directly maps (wrapper1) the schema of source1 to the mediation schema, say ChemicalML. The mediator uses some web service wrapper2 to adapt the output of source2 to the mediation schema. Finally, the last source knows about ChemicalML and doest itself the wrapping.

4 Active XML

We illustrate briefly how this may be put to work using Active XML (AXML in short), a language and a system developed at INRIA. A description of AXML may be found in [2].

The underlying model is based on AXML documents, i.e., XML documents possibly embedding calls to web services. More precisely, in AXML documents, some XML elements are interpreted as calls to web services. Such service calls may typically be used to acquire more data or to refresh data. An AXML peer consists of a repository that stores some AXML documents. These documents are active in that some of the web service calls may be automatically activated (in the style of active databases). An AXML

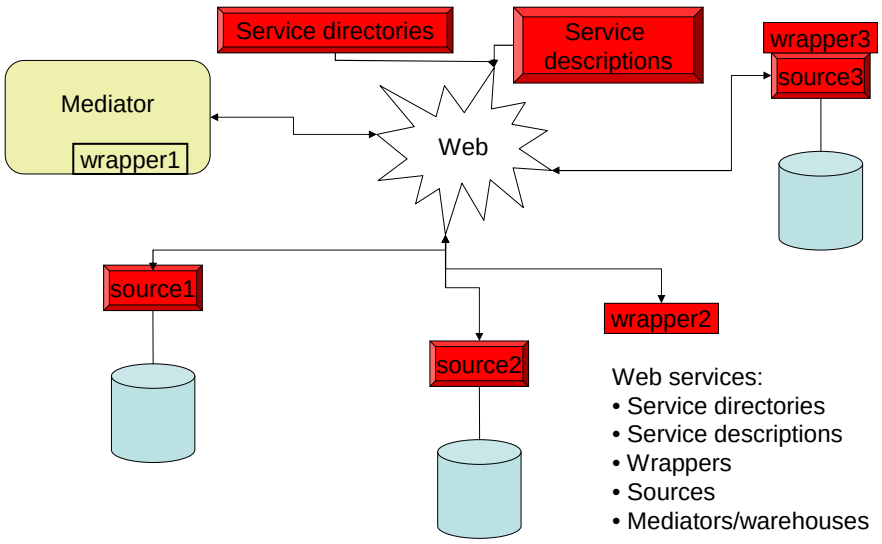

Figure 1: Data integration with web services

peer may also act as a server: Web services operations of the peer may be defined by means of queries (e.g., XQuery) on AXML documents. Being AXML data themselves, the arguments and results of service calls may also contain service calls allowing for distributed query processing over the web.

Data with embedded calls to operations is an old idea. (It is at the core of object databases [5] and has already been considered in the context of semistructured data in, e.g., [7].)

We will see that the combination of XML and web services (e.g., in AXML) provides a uniform vision of a very wide range of heterogeneous sources provided by web services. It also allows to capture different scenarios of data integration. Indeed, an essential question in this framework is when to actually invoke each particular service call. Based on the specification, the system may choose to do so when the information is needed (as in mediator systems), or beforehand (as in data warehouses). By allowing declarative control of service calls activation, the language allows to combine both approaches in a very flexible manner.

AXML documents may use standard web services that take XML data as arguments and return XML data. Beyond such simple web services, AXML data may refer to more complex services, which really allows to take fully advantage of the combination of XML and web service technologies:

Stream services: these are services such as subscriptions that entail a

possibly permanent relationship between the caller and the service. Such services provide the caller with a (possibly infinite) stream of answers/notifications and can be used, in particular, for change management. (Such features are not yet supported by the current version of SOAP, so they have to be handled on top of SOAP.)

Services exchanging AXML: Standard web services typically exchange "plain" XML data. In contrast, we allow web services to exchange AXML data that may contain calls to other services. This allows, for instance, for delegating portions of a computation to other sites, i.e., to distribute computations. (Recall that AXML uses an XML syntax, so there is no difficulty in exchanging AXML documents using SOAP.)

5 Simple scenario

In this section, we use a simplified syntax for AXML. The exact syntax (an XML dialect) would unnecessarily complicate the presentation. Suppose that we are at a site local.com and we want to integrate information about a particular chemical product "Pb". First, suppose that the sources that provide the relevant information are not known a-priori. Aa first phase is needed to discover them. This may be a call to a service directory:

```
<sc name="ChemicalDirectory" server"wci.com">
   <topic>Pb</topic> <format>ChemicalML</format> </sc>
```

The directory service may use the service descriptions and wrapper descriptions to compute the following answer:

```
<answer>
<sc name="source1" server="m1.com">
   <param> <sc name="wrapper1" server="local.com"> X </sc> </param>
</sc>
</query>
<query parameter="X">
<sc name="source2" server="m2.com">
   <param> <sc name="wrapper2" server="w2.com"> X </sc> </param>
</sc>
</query>
<query parameter="X">
<sc name="source3" server="m3.com">
   <param> <sc name="wrapper3" server="m3.com"> X </sc> </param>
```

```
</sc>
</query>
```

This says that for instance, for the second source, to get information about some keyword X, it suffices to

(using some UDDI web service), understand their semantics and possibly find or build the appropriate wrappers.

Consider first mediation. Suppose that we want to integrate source1, source2 and source3. They all provide web services. To integrate them, we need to use the tree wrappers.

It is simple to integrate the two sources using service calls to source1, source2 and w2to1. Note that this does not say much about optimization. Various optimization techniques may be used at the mediator site but that is a different issue. Observe that most such techniques involve some knowledge of the query support at source1 and source2, and of the capabilities of w2to1. The data is obtained in a lazy mode (only when needed). Such acquisition may be specified in AXML.

Now, consider a similar scenario in a warehousing mode. Data is prefetched. Suppose source source1 is providing an interface for change control, say a service call source1update. The warehouse subscribes to source1update and is then notified of changes to source1 that it propagates to its own repository. Now suppose that source source2 supports no such feature. Then the warehouse will have to query source2 regularly (get the full collection of data at source2) to maintain its repository. This may also be specified in AXML.

Finally, we could consider scenarios that would combine the two approaches. This is typically useful in practice. For instance, comparative web sites warehouse the prices of some products (such as houses) whereas they prefer to use a mediator approach for other products (such as plane ticket that change price very rapidly).

Conclusion In the presentation, the focus will be on combining XML and web services towards data integration. To show that, we will use the language (and system) AXML.

Acknowledgement We would like to thank all the people who worked on AXML and in particular, Bernd Amann, Angela Bonifati, Ioana Manolescu, Roger Weber.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann Publishers, San Francisco, California, 1999.

[2] Serge Abiteboul, Omar Benjelloun, and Tova Milo. Towards a flexible model for data and service integration, preliminary report,. In *Internat. Workshop on Foundations of Models and Languages for Data and Objects*, 2001.

[3] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocol (SOAP) 1.1. W3C Note, May 2000. `http://www.w3.org/TR/SOAP`.

[4] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, October 2000. `http://www.w3.org/TR/REC-xml`.

[5] R. G. G. Cattell, editor. *The Object Database Standard: ODMG-93.* Morgan Kaufmann, San Mateo, California, 1994.

[6] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Definition Language (WSDL) version 1.1. W3C Note, March 2001. `http://www.w3.org/TR/wsdl`.

[7] Ozone: Integrating Structured and Semistructured Data. T. Lahiri and S. Abiteboul and J. Widom. In *Proc. International Workshop on Database Programming Languages*, 1999.

[8] Universal Description, Discovery, and Integration of Business for the Web (UDDI).
`http://www.uddi.org`.

[9] The World Wide Web Consortium (W3C). `http://www.w3.org`.