The Nimble Integration Engine^{tm}

Denise Draper Alon Y. Halevy Daniel S. Weld

{ddraper, alon, dweld}@nimble.com
Nimble Technology, Inc.
1938 Fairview Avenue East
Seattle, WA 98102 USA

ABSTRACT

The consensus that XML has become the *de facto* standard for data interchange will spur demand for technology that allows users to integrate data from a variety of applications, repositories, and legacy systems which are located across the corporate intranet or at partner companies on the Internet.

In the past two years, Nimble Technology has developed a product for this market. Spawned from over a person-decade of data integration research, the product has been deployed at several Fortune-500 beta-customer sites. This abstract reports on the key challenges we faced in the design of our product and highlights some issues we think require more attention from the research community.

1. INTRODUCTION

XML's attributes of simplicity and flexibility have caused it to become the *de facto* standard for data interchange. Most data-management tool vendors have delivered some degree of XML support, and it is finally becoming possible to build a comprehensive data integration system without the expense and tedium of wrapper construction and maintenance. Although data integration has been studied by the research community for some time, we believe that the market for sophisticated data integration products is just beginning.

Nimble Technology's product provides a common interface to a corporation's myriad data sources, whether they are modern relational databases, legacy systems or structured files. The product, currently in beta, will launch this summer. We first present the product goals and architecture (Section 2), and then highlight several issues we think require more attention from the research community.

2. PRODUCT OVERVIEW

Nimble's typical customers include enterprises who need to conveniently query across multiple internal and external data sources. Market research suggests that the typical Global 2000 company has over one hundred enterprise applications running on fifteen platforms and eight data storage architectures[1]. While many of these sources are modern relational databases, legacy systems and structured files are surprisingly prevalant. This diversity of data sources is exacerbated by the current e-business trend which is driving companies to connect their online systems to those of their suppliers and channels.

The key feature distinguishing our product is the use of an XML-like data model at the system's very core. The first reason for employing XML is that it provides much greater flexibility in the kinds of data that can be handled by our system. The second reason, a much less technical one, is that users find data integration more compelling when XML is involved. Since XML is touted as a standard for data exchange within and across organizations, IT personnel find it easier to imagine applications of data integration when XML is the data transport format.

In discussions with potential customers we found many areas where users require data integration functionality. In one common scenario, the need for data integration arises when information about the customers of a company is scattered across multiple databases in the organization, and the company would like to learn more about its customers (by integrating all the data into one view) and to ensure that the data about customers is consistent across the databases. In some cases, the data sources have existed for a long time, and in others they have resulted from continuous activities of mergers and acquisitions. It is important to emphasize that in many of these cases, the option of creating a new unified data warehouse that stores all the information is not possible, either because of operational constraints, because of the cost of doing so, or for political implications within an organization (or across independent organizations such as suppliers).

Another class of applications for data integration is companies that need to build large-scale web sites which serve information from multiple internal sources. The task of building the web site itself is an enormous one in these cases, and it is very important to the customer to be able to separate the task of building the web site from the task of integrating the underlying data. Hence, they would like to provide the designers of the web site an already integrated view of their data sources.

2.1 System architecture

A diagram of the architecture of our system is shown in Figure 1. It is similar in spirit to the architecture of research prototypes (e.g., [4, 7, 5, 1], see [6, 3] for surveys), with the distinguishing factor that our system is built upon an XML

[1]Data from Gartner Group Report.

ACM SIGMOD 2001 May 21-24, Santa Barbara, California, USA

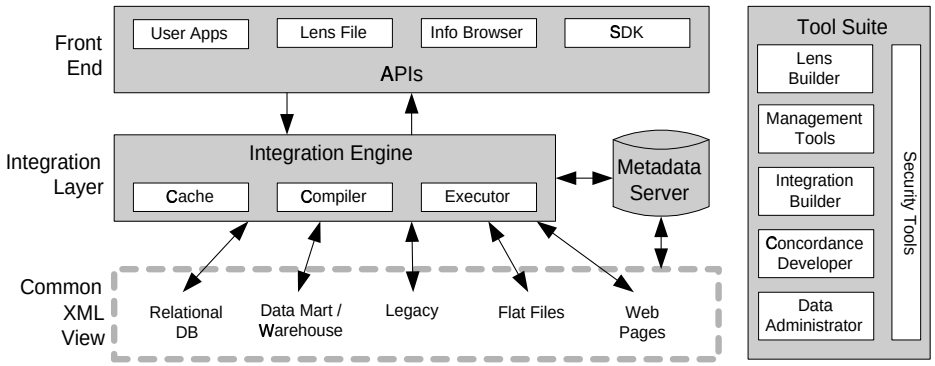

Figure 1: Architecture of product and ancillary tools.

data model.

We have adopted the World Wide Web Consortium's XQuery [2] language proposal, which consolidates the best elements of previous languages. Users and applications interact with the system using a set of *mediated schemas*. These schemas are essentially definitions of views over the schemas of the data sources (similar to the global-as-view approach [3]). It is important to notice that these schemas can be built in a hierachical fashion. That is, we can define successive schemas as views over other underlying schemas. This provides important flexibility when there are going to be multiple classes of users and applications using the system, and also facilitates defining the integration of the data sources, because it can be done in an incremental fashion (possibly in different parts of an organization).

The system *front end* is flexible, offering multiple layers of access. For example, a *Lens*(tm) is an object that contains a set of XML queries, parameters, XSL formatting, and authentication information. Result formatting can be targeted to specific devices (e.g., web interface, wireless device). Customers who wish to use a lower-level interface to the integration engine are also supported.

When a query is posed to the *integration engine* it is parsed and broken into multiple fragments based on the target data sources. The *compiler* translates each fragment into the appropriate query language for the destination source; for example, if an RDB is being queried, then the compiler generates SQL. Note that the compiler considers both the type of the underlying source, information concerning the layout of the data within the sources, and the presence of indices on the data. The *metadata server* contains the mappings that allow the query to be split apart and translated appropriately; mappings are set via the *management tools*.

Even though our main archiecture is built on a federated integration model, this alone is not always sufficient for all needs. Thus we support a compound architecture that includes offline data manipulation and replication as well, using our *data administrator* sub-system. In summary, our product has the following features:

- high-performance, scalable query processing of data from multiple sources
- dynamic data mapping and cleaning
- enable third party applications and devices to query and display results
- robust system managament
- XML as the unifying model underlying the system.

3. CHALLENGES

In the talk, we describe the key challenges we faced in designing our system, and point out several challenges that deserve more attention from the research community:

- Design of an underlying algebra to support operations on relational data as well as semistructured models, efficiently combine data from multiple models, and support a variety of optimization strategies.
- Development of a extensible data-cleaning framework that supports dynamic (query-time) cleaning when necessary and extension of concordance data bases.
- Incorporation of a flexible caching policy that allows a mixture of strategies, including both query-time access of fresh (operational) data and scheduled loading of data.

4. REFERENCES

[1] S. Adali, K. S. Candan, Y. Papakonstantinou, and V. S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proc. of SIGMOD*, pages 137–148, 1996.

[2] D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and M. Stefanascu. XQuery: A query language for XML. Technical report, World Wide Web Consortium, February 2001. Available from http://wwww.w3.org/TR/xquery/.

[3] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world-wide web: A survey. *SIGMOD Record*, 27(3):59–74, September 1998.

[4] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. *Journal of Intelligent Information Systems*, 8(2):117–132, March 1997.

[5] L. Haas, D. Kossmann, E. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proc. of VLDB*, Athens, Greece, 1997.

[6] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of PODS*, pages 51–61, Tucson, Arizona, 1997.

[7] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*, Bombay, India, 1996.