Data Integration: The Roles of XML and Database Middleware

Vanja Josifovski* IBM Almaden
Laura M. Haas† IBM Almaden

October 30, 2000

Today's life science companies face the problem of extracting and combining data stored in multiple large repositories. While complex inter-source queries are common in this domain, traditionally these are implemented by application code or left to the end-user to answer by combining the results of several simpler single-source queries. Furthermore, many queries require running computational algorithms for manipulation of sequences. We believe that the ability to query data in different repositories without regard to its location and representation can bring these companies to the next level of efficiency.

Database middleware systems offer users the ability to combine data from multiple sources in a single query, without creating a physical data warehouse. By "wrapping" the actual sources, they provide extensibility and encapsulation as well. The Garlic prototype [C+95] developed at IBM provides the user with a virtual database to which they can pose arbitrarily complex queries, though the actual data needed are stored or produced in several different sources. Garlic can compensate for function that may be lacking in a data source, to allow processing of these complex queries. Additionally, queries can exploit the specialized functions of a data source, so no function is lost in accessing the source through Garlic. The Garlic technology as incorporated in IBM's DB2 UDB product forms the basis of a services offering for the life sciences industry, known as DiscoveryLink [HKR+00].

The efficiency and correctness of the execution plans generated by a data integration system depend on the processing capabilities of the sources. For example, it is generally more efficient to process restrictions as close to the data as possible, to reduce the amount of data transferred. However, many Web-based data sources process requests using a form which only permits certain combinations of restrictions (e.g., a particular binding pattern). To form correct plans, the middleware must have information on both the data stored at the sources and the sources' capabilities.

XML is becoming a *de facto* standard for data exchange. Standards related to XML are being developed for defining data structure (schemas). These will in turn allow vertical standards for schemas within particular industries. While these standards will alleviate syntactic and semantic heterogeneity in the *data representations*, they do not provide means to describe the *processing capabilites* of the sources.

In practice it is not feasible to forsee all the possible types of data sources and classify them in the middleware system. Describing the capabilities of even simple sources is a daunting task which could require changes to the middleware code – severely limiting extensibility. Therefore Garlic adopts a *request-reply* framework where the middleware sends a request to the wrapper (representing a fragment of the user's query). The wrapper replies with a portion of the request which can be performed by the source, and the middleware will compensate for the remaining operations. This framework, also used in the ISO 9075-9 SQL/MED standard, allows the middleware to find an optimal processing plan without requiring description of the source capabilities in the middleware.

*vanja@almaden.ibm.com
†laura@almaden.ibm.com

The database middleware approach has a number of advantages. The paradigm described above allows the database query optimizer to find an optimal processing plan among the many possible ones, leading to execution times orders of magnitude faster than when plans are hard-coded in applications. For the users, the data in the sources appear to be in one virtual database and can be queried from existing applications. New applications can be developed by programmers unfamiliar with the actual data sources. Often, applications require local storage for (new) data used only locally. Rather than *ad hoc* solutions such as storing the new results in a file, this data can be stored in the middleware database where it can be queried easily. We want to combine these advantages with the strength of XML for standardizing data descriptions.

One step in this direction is to be able to query XML data sources from database middleware. To this end, we can build XML wrappers for DiscoveryLink. Several technical issues arise when writing a wrapper to interface an XML source with a relational middleware system. First, the wrapper needs to translate between the XML data model and the relational model used in the middleware. Next, the computational capabilities of the source must be available for use in the queries. Finally, many different data sources will export XML data. While it may not be possible to make one wrapper that works against all XML data sources, tools should exist to make writing a wrapper for an XML source as easy as possible.

We solve the problem of data translation via *table functions* that can be executed in the wrapper to produce tabular data from the XML input. The optimizer can choose when to apply these functions during query execution to achieve best query performance. An interface for passing information about costs between the middleware and the wrappers allows the optimizer to distinguish between alternative plans.

Computational capabilities of data sources come in many flavors. A source may provide functions that produce one or more outputs. DiscoveryLink allows a user to define *function templates* that are *mapped* to the functions in computational sources. These functions can then be used in queries without regard to the source of the arguments. For computations such as the BLAST [AGM+90] sequence matching algorithm that produce potentially large numbers of results, the computation may be modeled directly as a table by the wrapper.

Even XML sources storing data corresponding to the same schema may be accessed differently. For example, locally stored XML files may be accessed directly using the file system interface, while Web sources require a URL containing the form parameters. Therefore it is not possible to make a single general-purpose XML wrapper. Nevertheless, semi-automatic wrapper generators can read XML DTDs or XML Schema files and generate the needed SQL data definition statements, while a wrapper skeleton is completed with code for accessing the source by a wrapper writer. Further, existing wrapper code can be reused when developing a new XML wrapper for a similar source.

We believe that, with the features outlined in this abstract, DiscoveryLink provides the functionality required in the core of a life science information system. A combination of the XML tecnology with the Garlic data integration framework allows for rapid development of a system for processing complex queries over multiple different data and computation resources.

References

[AGM+90] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J Molecular Biology*, 215:403–410, 1990.

[C+95] M. Carey et al. Towards heterogeneous multimedia information systems. In *Proc. of the Intl. Workshop on Research Issues in Data Engineering*, March 1995.

[HKR+00] L. Haas, P. Kodali, J. Rice, P. Schwarz, and W. Swope. Integrating life sciences data – with a little garlic. In *Proc. BIBE*, November 2000.