A DATABASE FEDERATION PLATFORM FOR GENE CHIPS AND THE HUMAN GENOME DATABASE

B. Fu, S. Zhang, W. Chuang, and C. F. Dewey, Jr.
Massachusetts Institute of Technology, Cambridge, MA USA

Abstract- **Data sources that are typical of the next generation of biological information entities are gene chips that identify the individual genes in a given biological sample. These data are currently stored in a database format defined by the Genetic Analysis Technology Consortium (GATC). To interpret the chip data, we also need information about the genes themselves, as found in the Human Genome Database (HGDB). These two databases were conceived at different times to serve different purposes, and their designs differ significantly. Extracting information simultaneously from multiple databases has proved to be a very difficult problem.**

We have developed a system that will intelligently direct a single client query against a federation of databases. Our solution uses software standards common in the field today - XML, CORBA, and Java - but these standards by themselves are not sufficient. We have developed a new component called the Class Mapper, a software layer unique to each database. Each Class Mapper represents its database as an object-oriented schema consistent with the schema level of the federation. A Federation Platform reads the query, the Class Mappers execute the query across their respective databases, and the Federation Platform returns results to the client.

Keywords - **Biological databases, XML, CORBA, Java, database federation, genetic database, Class Mapper.**

I. INTRODUCTION

Recent advances in biology have produced an extraordinary number and variety of data sources that must be used together to address problems in medicine and biology [1-3]. Among the most visible are the data sources that capture our understanding of the human genome. Prominent among these is the Human Genome Database (HGDB) [4] that was designed to be a central repository for accumulating and disseminating genetic data. It contains three main types of information: regions of the human genome; maps of the human genome; and variations including mutations and polymorphisms. The size of the HGDB is more than a terabyte.

A second class of data sources comes from experiments. A representative of these data is the GATC format developed by the Genetic Analysis Technology Consortium [5] to capture the information in gene chips that contain 10^5 or more individual reaction wells, each of which represents an individual genetic test. A single gene chip entry in a GATC database would contain intensity information for each individual gene assayed by the chip as well as metadata giving the details of the experimental conditions and the analysis protocols.

We have taken the HGDB and GATC databases as representative of the many data sources that must be addressed together. Our objective is to be able to write a single query that will extract data from several databases simultaneously and in an intelligent manner. This means identifying the individual databases where specific data exist, parsing the query to address the right databases, and reassembling the result in such a way that, from the client point of view, the process is indistinguishable from a single query against a monolithic database.

The technical literature reveals a large number of attempts to federate databases using different methods and approaches. [6-14] This activity was very pronounced during the period around 1990-1994, but nearly all of those projects, including a promising effort called Pegasus at Hewlett Packard [13], seemed to disappear during the ensuing 5 years. In particular, the Pegasus project was to have users add remote schemas to be imported into the Pegasus database, thus making it a *dynamic* federated database. Non-object-oriented schemas were mapped to object-oriented representations within the global database. The global access language HOSQL (Heterogeneous Object SQL) had features of a multidatabase language system; however, local users were responsible for integrating imported schemas. All traces of this work vanished after 1993. Other notable approaches, including a system called MARGBench that originated at the Otto-von-Guericke University in Magdeburg, Germany [7], are discussed in the references.

The present work builds on previous efforts in our group to develop a common access technology between different databases. This approach came from the realization that writing and maintaining unique access software for each individual database would make the concept of federated databases nearly unworkable. The mechanism that was developed is called the Class Mapper [15-17]. The Class Mapper is a software layer that represents its underlying database to the federation as an object-oriented schema (a ClassMap) that completely describes the functionality of the underlying database.

Research sponsored by the Advanced Research Projects Agency.

Report Documentation Page

Report Date	Report Type	Dates Covered (from... to)
25OCT2001	N/A	-

Title and Subtitle
A Database Federation Platform for Gene Chips and the Human Genome Database

Contract Number

Grant Number

Program Element Number

Author(s)

Project Number

Task Number

Work Unit Number

Performing Organization Name(s) and Address(es)
Massachusetts Institute of Technology, Cambridge, MA

Performing Organization Report Number

Sponsoring/Monitoring Agency Name(s) and Address(es)
US Army Research, Development & Standardization Group (UK) PSC 802 Box 15 FPO AE 09499-1500

Sponsor/Monitor's Acronym(s)

Sponsor/Monitor's Report Number(s)

Distribution/Availability Statement
Approved for public release, distribution unlimited

Supplementary Notes
Papers from the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 25-28 OCT 201, held in Istanbul, Turkey. See also ADM001351 for entire conference on cd-rom.

Abstract

Subject Terms

Report Classification
unclassified

Classification of this page
unclassified

Classification of Abstract
unclassified

Limitation of Abstract
UU

Number of Pages
4

The Class Mapper also supplies the procedures to query the database in its own language and return results to the federation. Data exchange between the Class Mapper and the federation can then be carried out using reusable tools, and queries of one database appear to be operationally identical to the queries of any other database in the federation. Transport protocols can then be built using standards such as JDBC or CORBA to evoke TCP/IP. XML can also be used to define the data packages exchanged; this is discussed elsewhere [15].

II. The Federation Platform

The database federation architecture consists of a client, a Federation Platform with a local database, and two or more external databases that contain information used to compose an answer to the query from the client. Fig. 1 presents an overview of the transaction architecture. We have used the problem of federating the GATC database and the HGDB database in this example. Numbers in circles indicate the steps in processing a query that initiates at the client at point ①. The Federation Platform is designed to parse the query, communicate with the Class Mappers on the individual databases, and create and maintain a transient database (the Local Database in Fig. 1) to efficiently extract specific information to satisfy the query.

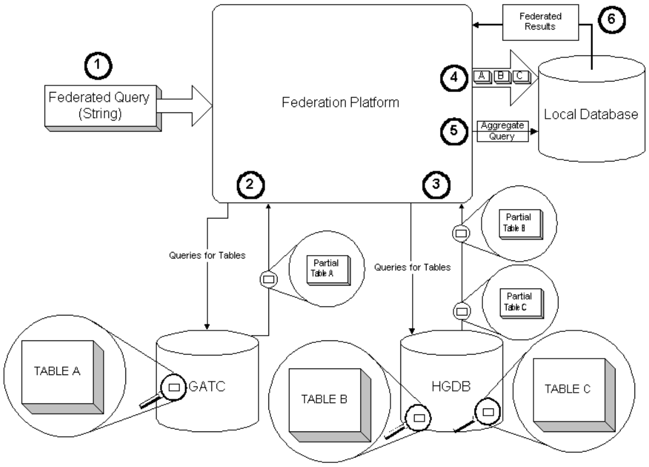

Fig. 1 The transaction architecture in a federated database.

Steps ② and ③ represent separate queries to the GATC and HGDB databases to retrieve individual parts of the information that will be used to satisfy the query in ①. What is returned to the Federation Platform is a series of partial tables from the GATC and the HGDB. These partial tables are then used in ④ to construct a Local Database that contains only information relevant to the current query. The fact that these tables are written back to the Federation Platform upon each query insures that the data used in answering the query are current. The actual query and other processing algorithms are run against the Local Database in ⑤, returning the result ⑥. At the end of the query, the information in the Local Database is discarded.

By using a Local Database, all of the results that have been retrieved from the remote databases are easily accessible using standard database tools that are designed for such purposes. Storage scalability is a second advantage. Database systems are built to be scalable to terabytes with good performance. With good commercial database technology, one can scale to multi-gigabyte file queries without severe performance penalties. Without using database functionality, such file sizes could quickly exceed the capacity of system memory, forcing the use of brute-force memory swapping to handle complex queries. Finally, the Local Database can take advantage of built-in database management features such as query optimization, security, recovery, and data integrity.

Although we have yet to implement it, one could also make use of local cache mechanisms to allow partial tables from the external databases to be saved between queries so that building sequential Local Databases from the same partial tables could be accelerated. A mechanism much like the cache control used on internet browsers could be adopted. This could also reduce network traffic in cases where the partial tables were large and/or the network connections were slow.

III. Implementation

Our first implementation of the GATC-HGDB database federation was implemented on a Sun 450 running Solaris 7 using local disks and separate databases representing the GATC and HGDB databases. We built all three databases - the GATC database, the Local Database, and the representative HGDB database - using Informix Dynamic Server 2000. For the HGDB database, we did not use the entire database at the NIH because it is over a terabyte in size. The difficulties of replicating and keeping current such a large local database would have detracted from our main goal of achieving feasibility testing. A partial HGDB database (we call it "HGDB Lite") was therefore built locally to handle information specifically relevant to the test queries that we were writing. It was built to conform to the full HGDB Class Map as defined by Chuang [17]; we simply populated a small fraction of the available tables.

Class Mappers were available for both the GATC and the complete HGDB databases from the work of William Chuang [17]. The GATC database was populated with about 500 sets of Affymetrix gene chip data that were kindly supplied by Michael Cardone of the MIT Department of Biology. No Class Mapper was required for the Local Database because it was created on the fly by the Federation Platform itself and therefore was completely known to the program. The "HGDB Lite" had about two dozen tables and roughly a

GByte of representative data that covered a range of queries related to GATC information.

The DNA identifiers (called Accession ID's) in the GATC database correspond to DNA segments with genomic characteristics. However, these characteristics are stored in the tables and connections of the HGDB. In order to associate a particular DNA segment from an experiment with its genomic characteristics, the two databases must be used in parallel. The problem then becomes the task of querying data across the two database domains.

Merging the two databases does not represent a feasible solution to the problem. Both databases are dynamic because new experimental data are being continually added to the local GATC database and the HGDB is itself an evolving entity. As our prototype demonstrates, federation offers a viable solution that can be applied to multiple databases and multiple queries.

As might be expected from the design shown if Fig. 1, the Federation Platform encompasses a significant amount of functionality. First, the ClassMap for each database of the federation is stored in the Federation Platform and used to construct a global description of the entire federation called a *ClassMapRepository*. A hash table is used for fast lookups of table-to-database mappings and conflict resolution in case more than one database has the same table name.

From this global description, a series of data structures are created: these structures define the appropriate means of querying and accessing any individual piece of data in the entire federation. The details of this design can be found in the Thesis by Ben Fu [18]. Original queries from the client are parsed against this global description by the Federation Platform, and the relevant data are automatically queried from the selected tables in the several databases. The original query structure is then used to create the Local Database and a final query is made against the Local Database to produce the results that are returned to the client.

In this prototype, extensive use was made of JDBC to query the databases, construct the Local Database, and handle the table objects. This was convenient because Informix supports a Type 4 (native) JDBC driver. The object support within Java and JDBC also simplified many of the internal structures [19]. It is entirely consistent with this architecture to use other methods, including ODBC and CORBA, to create the internal data structures and carry out the communication with the Class Mappers. We have done so on other related problems with good success.

IV. Discussion and Conclusions

This paper has presented a new approach to database federation that we believe will be useful in the coming decade of bioinformatics. The amount of biological data that is being created is staggering. The human genome is only the beginning. New databases are being constructed for different species, for different classes of proteins, and for molecular pathways. Coupled with these data are new experiments that yield protein information, genetic information and gene fragment data. The task of creating paths back and forth between these different data sources is truly daunting.

Our short-term goal has been to pick a single pair of important databases and demonstrate an architecture that can be used to complete cross-database queries using standard database tools and programming methodologies. Our results are readily generalizable to other experimental methods and other database collections. No compromises have been made that would restrict this architecture from scaling to very large databases and large numbers of individual data sources. An extended scheme is shown in Fig. 2.

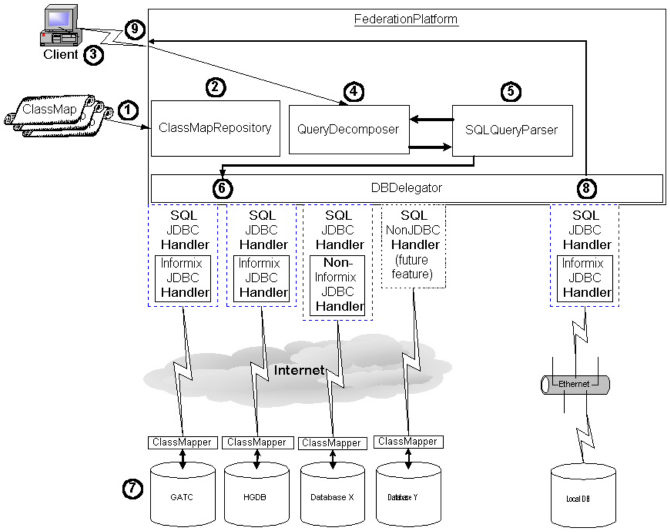

Fig. 2. The general Federation Platform architecture.

Acknowledgment

The research presented in this paper includes contributions of several earlier members of our research group, and is only partially represented by the references to their publications. We wish to thank Ngon Dao for his continuing guidance on the architecture of the federation methods, Patrick McCormick for continuing help with the development of Class Mappers for new applications, and Paul Ferraiolo for his consultation on best practice software for bioinformatics from a commercial point of view.

References

[1] J. Macauley, H. Wang, and N. Goodman, "A model system for studying the integration of molecular biol-

ogy databases," *BioInformatics Journal*, Vol 14, No. 71 pp 575-582, 1998.

[2] N. Goodman, S. Rozen, and L. Stein. "The case for componentry in genome information systems." *Meeting on Interconnection of Molecular Biology Databases*, Stanford University, 1994.

[3] S. K. Moore. "Harmonizing data, setting standards." *IEEE Spectrum* vol. 38, issue 1, pp. 111-112. January 2001.

[4] Human Genome Project. Report of the invitational DOE Workshop on Genome Informatics. http://www.ornl.gov/hgmis/publicat/miscpubs/bioinfo/inLrep2.html. April 1993.

[5] http:/www.gatconsortium.org. Entries to this web site are older than references given in the commercial web site of Affymetrix: http://www.affymetrics.com.

[6] J. Kohler, M. Lange, R. Hofestadt, S. Schulze-Kremer. "Logical and semantic database integration." *Bio-Informatics and Biomedical Engineering,* pp. 77-80, *Proceedings of IEEE International Symposium* on Nov. 8-10, 2000.

[7] A. Freier, R. Hofestadt, M. Lange, and U. Scholz. "MARGBench – An aproach for integration, modeling and animation of metabolic networks." *Proceedings of the German Conference on Bioinformatics*, Hannover, 1999.

[8] P. D. Karp. "A Strategy for Database Interoperation." *Journal of Computational Biology*, vol. 2, pp. 573-586, 1995.

[9] B. Reinwald, H. Pirahesh, G. Krishnamoorthy, G. Lapis, B. Tran, S. Vora. "Heterogeneous query processing through SQL table functions. " *Proceedings of the 15th International Conference on Data Engineering*, pp. 366-373, 1999.

[10] G.J.L. Kemp, N. Angelopoulos, P.M.D. Gray. "A schema-based approach to building a bioinformatics database federation." *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering*, 2000.

[11] R. J. Robbins. "Bioinformatics: Essential infrastructure for global biology." *Journal of Computational Biology*, vol. 3, pp. 465-478, 1996.

[12] O. Jautzy. "Interoperable databases: a programming language approach." *Proceedings from IDEAS '99 International Symposium on Database Engineering and Applications*, pp. 63-71, 1999.

[13] A.R. Hurson, M.W. Bright, S.H. Pakzad. *Multidatabase Systems: An Advanced Solution for Global Information Sharing.* IEEE Computer Society Press, 1994.

[14] A. Elmagarmid, M. Rusinkiewicz, A. Sheth. *Management of heterogeneous and autonomous database systems.* Morgan Kaufmann Publishers Inc., 1999.

[15] P.J. McCormick. *Designing object-oriented interfaces for medical data repositories*. M. Eng. Thesis, MIT. 1999.

[16] N. Dao, P.J. McCormick, C.F. Dewey, Jr. "The human physiome as an information environment." *Annals of Biomedical Engineering.* vol. 28, pp. 1031-1042, 2000.

[17] W. Chuang. *Design of a genetics database for medical research.* M. Eng. Thesis, MIT. 2000.

[18] B. Fu. *Design of a genetics database for gene chips and the human genome database.* M. Eng. Thesis, MIT. 2001.

[19] C.S. Horstmann, G. Cornell. *Core Java 2, Volume 1: Fundamentals.* Prentice Hall PTR/Sun Microsystems Press, 2000.