



BioQuery: an object framework for building queries to biomedical databases

James M. Brundage and Christopher Dubay*

Division of Medical Informatics and Outcomes Research, BICC Oregon Health & Science University, 3181 SW Sam Jackson Park Rd, Portland, OR 97239, USA

Received on September 10, 2002; revised on November 5, 2002; accepted on November 13, 2002

ABSTRACT

Summary: BioQuery is an application that helps scientists automate database searches. Users can build and store queries to public biomedical databases, and receive periodic updates on the results of those queries when new data is available. The application is implemented on a portable object framework that can provide database-searching capability to other applications. This framework is easily extensible, allowing users to develop plug-ins that provide access to new databases. BioQuery thus provides end-users with a complete database searching interface and updating service, and gives developers a toolkit to provide database-searching capability to their applications.

Availability: Free to all users: <http://www.bioquery.org>

Contact: support@bioquery.org

INTRODUCTION

The acquisition and publication of biological data is characterized by accelerating growth, in which every year sees an increase in the amount of data generated. This increasing flow of data has been managed by the creation of numerous biomedical databases coupled to search engines, query interfaces, and analysis tools (Gelbart, 1998; Baxevanis, 2002).

One of the largest repositories of data and analysis tools can be found at the National Center for Biotechnology Information (NCBI; <http://www.ncbi.nlm.nih.gov/>; Wheeler *et al.*, 2001). Despite the power and ease of use of these tools, they do not meet all of the needs of biological researchers (Andrade and Bork, 2000; Chu *et al.*, 2000). One problem is that databases continually receive new content, but few provide a means of automatically re-querying the updated material. Researchers may want to invest considerable time developing complex queries that return a very selective subset of data, so long as they only have to input such a query once and have it periodically resubmitted (Andrade and Bork, 2000; Chu *et al.*, 2000).

This paper describes the BioQuery program and its

underlying Query object framework. BioQuery is a Java application that provides end users with an interface for building queries and receiving periodic updates from public biomedical databases. The Query object framework hides and encapsulates the database-specific details, making the program easy to extend and maintain as databases change.

ARCHITECTURE OF THE QUERY FRAMEWORK

The Query framework models the way scientists build queries, rather than how those queries interact with a specific database. At the core of the framework is the Query class (Fig. 1), which models a user-created query and tracks its submission. The Query class does not contain database-specific details, which are instead placed in a human-readable XML document. Creation of new Querys is the responsibility of the QueryFactory class, which reads the XML document and returns a correctly configured Query object for any available database.

The submission of a Query to its database involves a number of specific implementation details that cannot be generalized. This responsibility falls on the abstract QuerySubmitter class and its concrete subclasses, which act as a Bridge between the generic Query class and the specific implementation of each database (Gamma *et al.*, 1995). Subclasses of QuerySubmitter must translate the structure of a Query into a format the database can understand, submit the query over the Internet to its database, and parse and return the results. The Query framework thus handles the problem of database interoperability by encapsulating submission details into one class per target database. The framework currently provides access to eight databases, and new databases can be added in two steps: add a new entry to the XML file, and create a new subclass of QuerySubmitter. The existing classes do not need to be modified or recompiled because the new class will be dynamically added at runtime. If the interface to an external database changes, only one class needs to be modified to fix any problems that might arise.

*To whom correspondence should be addressed.

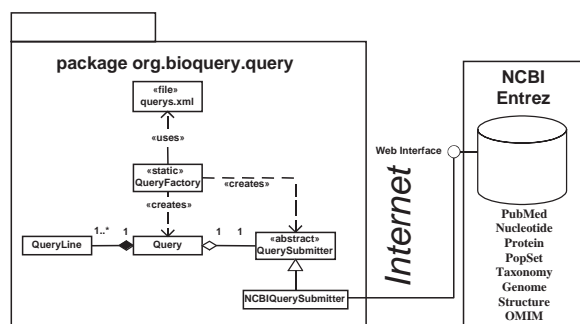


Fig. 1. Overview of the Query framework. The Query class is generic and does not contain database-specific details. Database-specific information is contained in the *query.xml* file, which is read by the QueryFactory class the first time it is invoked. QueryFactory creates individual instances of Query objects and loads them with database-specific information. The QueryFactory also gives each Query a specific implementation of QuerySubmitter that contains all of the code required to contact the database, submit the query, and return the results. Not all classes are shown for clarity.

The Query framework has few external dependencies and is currently configured to search eight biomedical databases from the NCBI. It can thus be used as a toolkit for providing database searching capability to other programs. We have created one such program: the BioQuery application.

THE BIOQUERY APPLICATION

BioQuery is a complete end-user program that allows scientists to automate database searches. BioQuery is for researchers who want to extract a selective subset of biomedical data and keep it up-to-date, or who want to be notified of new discoveries. An installation program loads BioQuery and all required components. BioQuery launches as a stand-alone program with a graphical user interface (GUI) and built-in tutorial and help pages. End-users can build searches to eight different databases at the NCBI using textual search terms and nested Boolean operators. A list of available search fields and the ability to build searches line by line aid query construction. Users can edit any part of a saved search strategy, allowing gradual refinement over time. Saved searches can be set to automatically resubmit themselves at various intervals, and return only those items that have been entered or modified since the last automatic search. The data can be e-mailed to the user or saved as a file.

The BioQuery program is a simple client-server application. The client portion provides the user interface to build searches, while the server holds saved searches and checks for new data nightly. Because BioQuery is built

upon the Query framework, it is extensible. A developer can add additional databases to the framework as plug-ins, and the BioQuery application will detect this at runtime and make them available. The program synchronizes application files between the client and server to simplify distribution of new versions of the software. If a database changes, the program can be modified on the server and client programs will automatically detect this change and ask the user for permission to update the files on the client, providing uninterrupted service.

CONCLUSIONS

There is a great deal of interest in integrating biomedical databases, particularly through the use of Web Services (Stein, 2002). Although this will simplify the construction of tools, there will still be a need for end-user applications to provide querying and analysis services. The BioQuery application and the Query framework demonstrate how object modeling can be used to build such a tool. The Query framework is a component-based toolkit that provides the ability to create, save, and submit database queries. The BioQuery application is a full service end-user program that helps users create and store searches, and receive periodic updates on their search results. BioQuery is an open source project: extensions and modifications are encouraged, and the source code and detailed documentation are available through the website. The application is freely available to everyone, and can be installed from <http://www.bioquery.org>.

ACKNOWLEDGEMENTS

This work was supported by NIH-BISTI grant no. T15-LM07088-10.

REFERENCES

- Andrade, M.A. and Bork, P. (2000) Automated extraction of information in molecular biology. *FEBS Let.*, **476**, 12–17.
- Baxeavanis, A.D. (2002) The Molecular Biology Database Collection: 2002 update. *Nucleic Acids Res.*, **30**, 1–12.
- Chu, W.W., Johnson, D.B. and Kangaroo, H. (2000) A medical digital library to support scenario and user-tailored information retrieval. *IEEE Transactions on Information Technology in Biomedicine*, **4**, 97–107.
- Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, pp. 151–161.
- Gelbart, W.M. (1998) Databases in genomic research. *Science*, **282**, 659–661.
- Stein, L. (2002) Creating a bioinformatics nation. *Nature*, **417**, 119–120.
- Wheeler, D.L. et al. (2001) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.*, **29**, 11–16.