# QueryDesigner: Simplifying Access to Remote Scientific Databases

Mark Newsome and Cherri Pancake {newsome|pancake}@cs.orst.edu)
Department of Computer Science
Oregon State University
Corvallis, OR 97331 USA

Joe Hanus (hanusj@bcc.orst.edu)
Department of Botany and Plant Pathology
Oregon State University
Corvallis, OR 97331 USA

***Abstract:*** Many large-scale scientific data collections are stored in relational databases that use command-line SQL as the query interface. For years, scientists and other occasional users have had a difficult time accessing their data because SQL is terse and unforgiving of errors.

With the advent of the Web browser, forms-based interfaces have been layered on top of relational databases, shielding the end-user from the complexity of SQL. While these interfaces are easy to use, building them requires programming experience. The increasing demand for additional database interfaces is often not met due to declining personnel budgets and a shortage of programming staff.

Our solution was to build a user-oriented tool called QueryDesigner that enables non-computing experts to explore, construct, and personalize query interfaces from a Web browser. QueryDesigner is a Java applet that eliminates the requirement that the interface designer understand SQL, HTML, or any specialized language.

QueryDesigner was built in collaboration with scientists who actually work with large-scale biological databases. It was designed to allow the scientist to easily compose forms-based interfaces that support the way they search databases. QueryDesigner helps the user understand unfamiliar databases by acquiring information on database organization and presenting it as interactive diagram. The applet transparently composes SQL queries, issues requests to a remote database, receives and formats the results.

## Scientists Have Special Requirements

Unfortunately, scientists — who are often not computer experts — do not have an easy time retrieving information stored in SQL databases. To formulate an SQL query, the user must first look up the names of tables and fields, which requires the use of a proprietary (non-standard) tool (e.g., Oracle's **sqlplus**). Moreover, SQL itself is terse, error prone, and unfriendly. It is easy to misspell a keyword or data name, omit a "join" clause, or construct a query that returns unintended results [Sme95]. There is also a lack of meaningful feedback during the query process, despite the fact that it is unlikely a casual user will be able to formulate a query correctly on the first attempt [JV85]. The user bears the burden of determining what went wrong, based on cryptic messages that reveal little about the source of the problem.

We were able to capitalize on what we learned about forms- and hypertext-based query interfaces for biological researchers during a previous interdisciplinary project [HNPM95] where we analyzed how users interact with biological databases [New96]. Scientists in the biological disciplines typically begin searches with only a partial idea of where they are going and what kind of results might be available. It is crucial for them to be able to explore the data and investigate the relationships between database entities, without having to formulate complex SQL queries.

With incomplete criteria, a search is likely to return a large result set; most database software simply returns all results in a rapidly scrolling display. Scientists need to be able to determine the size of the results before they are returned, in order to know whether the query needs refining. Support is needed for retrieving results in incremental blocks, paging through them, as well as saving the data in spreadsheet format. Moreover, keyword search is essential in scientific databases because they often contain unstructured data such as field notes or historical annotations.

Another common characteristic of biological data is the inclusion of a large number of attributes; no single organism is likely to contain values for every attribute, so database records contain a large number of empty (or null) data fields. This makes it difficult for users to find pertinent information, since current database management software provides no means of suppressing empty ("NULL") output.

We also found that a number of unsuccessful queries were caused by users' misspelling long scientific names, orthographic and typographic errors in the database itself, or the use of discipline-specific naming variants. When users must type values into empty blanks on a form, specifying search criteria becomes a guessing game. When no results are returned from a query, users are left wondering whether there really is no data available, or if their input was incorrectly specified.

## How Query Designer is Used

The QueryDesigner applet (see http://www.nacse.org/qd) employs a graphical interface that permits the end-user to either personalize or create a forms-based query interface to a remote database. The tool eliminates the need for the end-user to learn SQL or proprietary scripting languages. Instead, the user formulates queries implicitly, by specifying on an interactive E-R diagram which data fields should serve as inputs and outputs. QueryDesigner checks to ensure that the joins are legitimate and that all database elements are joined, preventing several of the most common semantic errors. Once the interface elements have been specified, the tool generates an initial query interface, using default rules for placing elements; this can be edited by the user.

In designing a query form, the user can specify whether an input field is to appear blank, pre-initialized to a constant, or shown as a query-list. The query-list feature eliminates user guesswork and spelling mistakes on the query form, by pre-fetching a list of allowed valued from the database. This is particularly useful for scientific databases, given that taxonomic and other scientific names tend to be long and easily misspelled. Although keyword searching is not supported directly by SQL, our software permits the scientist to search for keywords, even when it is not known in advance which fields should be searched.

When submitting a form, QueryDesigner automatically composes an SQL query, establishes a connection to the database, receives the results, and displays them. By default, QueryDesigner automatically suppresses the printing of empty (or NULL) fields to prevent screen clutter. A pull-down is used to limit the number of results returned at a time. A slider at the bottom of the screen permits the user to scroll through results one record at a time. Clicking the SAVE-AS-SPREADSHEET button permits the user to save results in a form accepted by most spreadsheet software.

Finally, QueryDesigner offers advantages in terms of its architecture. Accessible from a Web browser, QueryDesigner is platform independent as well as location-independent of the underlying database. Perhaps the most important advantage is that QueryDesigner is independent of any particular DBMS software. Through JDBC, QueryDesigner supports most relational databases that are addressable over the Internet.

## References


[HNPM95] Joe Hanus, Mark Newsome, Cherri Pancake, and Larry Moore. Facilitating access to microbial germplasm through electronic networks. *Proceedings of the 5th International Conference on Pseudomonas syringae Pathovars and Related Pathogens*. Berlin, Germany, September 1995.

[JV85] Matthias Jarke and Yannis Vassiliou. A framework for choosing a database query language. *ACM Computing Surveys*, 17(3):313-340, September 1985.

[New96] Newsome, M. A Browser-based Tool for Designing Query Interfaces to Scientific Databases. Ph.D. Dissertation, Dept. of Computer Science, Oregon State University, 1996.

[Sme95] John B. Smelcer. User errors in database query composition. *International Journal of Human-Computer Studies*, 42:353-381, 1995.



**Acknowledgements**

QueryDesigner was developed as part of a joint project of the Microbial Germplasm Database (MGD) and the Northwest Alliance for Computational Science and Engineering (NACSE). MGD is funded by the National Science Foundation (BIR 9503712). NACSE is a Metacenter Regional Alliance sponsored by NSF grant ASC 9523629. The Java version and extensions were sponsored by a NAVO Major Shared Resource Center PET project.