TableMaker: An ad hoc Query Tool for Relational Databases

C. Lushbough1, J. Duwick2, Q. Dong3, D. Jennewein1, J. Reynoldson1, and V. Brendel4
1 Department of Computer Science, University of South Dakota, Vermillion, SD 57069, USA
2 Department of Genetics, Development and Cell Biology, Iowa State University, Ames IA 50011, USA
3 Center for Genomics and Bioinformatics, Indiana University, Bloomington, IN 47405, USA
4 Department Genetics, Development and Cell Biology, and Department of Statistics, Iowa State University, Ames, IA 50011, USA

Abstract: Most Web servers hosting biological data limit users to a defined set of search options and output formats that are short of the whole range of options available to users with direct database access. However, to make full use of the wealth of data in the database resource, it is desirable to have an intermediate solution that provides a broad range of flexible query and output options through a Web portal.

TableMaker is a generalized Web-based ad hoc query tool that connects to relational databases using the Java Database Connectivity (JDBC) API. It allows researchers to build sophisticated queries and select data items for display in tabular format through a menu-driven query building interface, without requiring users to type Structured Query Language (SQL) statements. A query wizard is included for additional ease of use.

Availability: TableMaker is available through SourceForge at https://sourceforge.net/projects/tablemaker.

Contact: Carol.Lushbough@usd.edu

Keywords: Ad Hoc Query, Relational Database.

1 Introduction

The number of molecular biology resources available on the Internet is growing rapidly, with 960 public databases indexed in 2007 [1] compared to approximately 260 in 2000 [2]. These database resources store vast amounts of diverse data. Such data includes genetic and genomic data, molecular structures, metabolic pathways, and gene expression data. In order for biologists to access the information stored within these databases, web-accessible Graphical User Interfaces (GUIs) are typically provided in the form of text-input fields, checkboxes, selection menus, etc., used for constructing queries. Designing online query tools for a broad clientele often involves compromises between the clarity of the interface and the complexity of the underlying database. While meeting the needs of the majority of researchers, such query tools may be inadequate for some. For example, users of GenBank’s Entrez sequence data search tool [3] (http://www.ncbi.nlm.nih.gov/) might want to access information regarding “tissue type”, e.g., “retrieve all the EST sequences from maize leaf tissue” for studying gene expression patterns. Despite the fact that “tissue type” information is stored in GenBank, it cannot be accessed online because “tissue type” is not one of the designated search fields.

Another limitation of many interfaces to data resources is the lack of ability to generate flexible tabular reports from query results. Similar to the query limitation described above, the available output options are also usually fixed by the Web designers, whereas users may want to specify what attributes of a particular record should be included in the output. In addition, the query output format may be inconvenient for data export and parsing for further analysis.

One option for overcoming these limitations is to allow direct access to the backend database system. However, this solution is not practical for researchers who are not familiar with the internal database language (typically SQL). Therefore, a generalized but easy-to-use solution is desirable. We present TableMaker, an ad hoc web-based query tool that connects to any relational database and provides a simple but powerful interface for database queries and data display. With TableMaker, researchers are able to build sophisticated queries and select data items for display without being required to know SQL. Usability is further enhanced by means of a customizable query wizard.
2 Methods

2.1 TableMaker design and function

All relational database management systems (RDMS) use built-in metadata – table, column, index and key definitions to help describe the underlying database. When TableMaker initially connects to a relational database, it uses the JDBC API to retrieve the database’s metadata and store it in the TableMaker database. Figure 1 depicts a simplified schema for this database. The data in the TableMaker database is used to populate the GUI controls on the TableMaker query screen, define the user views of the underlying database, and construct SQL queries. Once stored, this information can be modified in order to customize the user’s view of the underlying database. For example, actual names as well as aliases are stored for each table and column belonging to the underlying database. These aliases are the values that appear in the GUI and can be set to user-friendly values. Additionally, if an alias is set to “HIDE” then the associated table or column will not be presented to the user. This is particularly useful for “hiding” cross-reference tables. If foreign keys (attributes in a table that refer to the primary key of a record in another table) were not explicitly defined in the underlying database when the tables were created, relationships can be defined in the TableMaker database by inserting records into its relationship table.

The standard TableMaker interface provides dropdown menus for selecting query tables and query variables available for each table (see Figure 2). Additional query delimiters can be added with choice of Boolean operators. Output fields are then selected from another set of dropdowns. If two or more tables are used for output, the user can specify whether TableMaker performs a left outer join (return all rows in which a match occurs) or an inner join (return only non-null rows). Search results appear in tabular form, along with additional options for saving the data as a text file or saving the query itself for later use or modification (see Figure 3).

The actual SQL query statement generated by the TableMaker system is provided in a text box above the search results facilitating documentation or modification of the query. More advanced users are able to enter and execute the SQL statement directly from this text box. The system does not permit execution of SQL commands that would result in modifying the underlying database. Examples of these commands include DROP TABLE, INSERT INTO, and DELETE FROM.

TableMaker is built entirely with freely available, open source technologies. The TableMaker software is a Java Server Pages (JSP)/Java Web application utilizing Java 5 running inside an Apache Tomcat (http://tomcat.apache.org/) servlet container. The relational database backend is implemented with MySQL (http://www.mysql.org/). Installation involves creating the necessary metadata databases and then deploying TableMaker to the servlet container. SQL scripts for populating the metadata databases as well as a pre-compiled Web application archive are provided in the distribution. Once deployed, TableMaker’s Web application configuration file must be updated with the information required to connect to the target database.
Fig. 2 TableMaker query page: The query page allows users to create queries by selecting from lists of available tables, related tables and columns. The data to be displayed is determined by selecting for a list of available tables and columns. If two or more tables are used for output, the user can specify whether TableMaker performs a left outer join (return all rows in which a match occurs) or an inner join (return only non-null rows).

2.2 Query wizard

The TableMaker query wizard was created to help researchers learn to construct ad hoc queries against the underlying database. The wizard walks the researcher through the process of selecting the correct tables to search, specifying the correct search criteria, and selecting the desired attributes for display by presenting a series of increasingly more specific questions. Through the use of a decision tree classification system, the wizard starts from a set of common general “requests” (e.g., “I am interested in ESTs”) and based on the user’s selection, leads the user down a specific branch of the tree and presents a set of common sub-requests (e.g., “I am interested in tissue types”).

The query wizard decision tree is represented by a directed acyclic graph (DAG) and is stored in a MySQL database. It is implemented through a series of JSPs that facilitate interaction with this graph. The inner nodes of the graph are the various questions to be presented to the user, and the leaves are the terminal questions that carry the user over to TableMaker.

Interaction between users or administrators and TableMaker is facilitated through the use of Asynchronous JavaScript and XML (AJAX), which enables the query wizard, for example, to take the data related to the question being posed by the user and populate the GUI in the TableMaker query screen. Wizard-guided queries lead to a pre-populated, standard TableMaker interface, which allows users to modify a query before actually executing it.

Fig. 3 TableMaker results page: The results page displays the results of a query in tabular format. It also provides the ability to download the results and execute subsequent SQL commands. Buttons are provided for sorting by column.

Fig. 4 TableMaker query wizard: The query wizard was created to help researchers learn to construct ad hoc queries.
queries against the underlying database. The wizard walks the researcher through the process of selecting the correct tables to search, specifying the correct search criteria, and selecting the desired attributes for display by presenting a series of increasingly more specific questions.

3 Results

3.1 PlantGDB instance of TableMaker

The TableMaker tool has been implemented with PlantGDB, a web-based database of plant molecular sequences that are downloaded from GenBank and reorganized for analysis and retrieval using MySQL [4]. TableMaker was connected to PlantGDB in order to provide researchers with robust searching and reporting capabilities (http://www.bioextract.org/genbank/home/index.jsp; all examples discussed in the text are demonstrated at this site). With the TableMaker tool, researchers can execute queries that are not possible to execute at GenBank (e.g. “show me all the plant promoter sequences” or “show me all the species names in the family Poaceae”). Figure 1 demonstrates an example of using TableMaker at http://www.plantgdb.org to retrieve all the annotated long terminal repeat (LTR) sequences found in plant retroelements. Because all the necessary information for locating LTR is stored in the backend PlantGDB database, a few clicks on TableMaker at PlantGDB will execute the corresponding SQL statements and obtain the database-query results easily.

3.2 Other TableMaker Usage

To demonstrate the applicability of TableMaker to diverse databases, we have implemented TableMaker versions of the MINT and Panzea databases for protein-protein interactions and comparative genomics, respectively [5, 6]. These TableMaker instances can be accessed at http://biota.usd.edu/mint/ and http://biota.usd.edu/panzea/.

4 Future Work

The TableMaker is a generalized Web-based ad hoc query tool that is designed to help users easily query a relational database. There are several areas within the TableMaker system that we would like to expand. Currently, the system is primarily designed to connect to MySQL databases. Because the TableMaker system uses standard the JDBC API to connect to the underlying database, it is able to connect to other Database Management Systems (DBMS) such as PostgreSQL or Oracle but additional functionality needs to be added to take advantage of specific DBMS characteristics.

The queries in TableMaker could be expanded to include other table joining mechanisms such as right outer join and cross join to enrich the functionality of the current inner and left join operations. Providing a graphical view of the underlying database (i.e. simplified schema) might also enrich to the TableMaker by helping the user understand the structure of the data.

Adding an administrative module for defining views of the system would be another addition that would make the system easier to use. This module could provide a GUI for defining aliases, hiding elements, and defining table relationships.

Lastly, it is very easy in SQL to construct queries that return a huge amount of data, such as when querying large tables, or when tables are joined without giving appropriate join conditions. These types of queries have the potential to bring down a database server. Although the TableMaker system has a “time-out” mechanism to terminate queries taking an excessively long period of time, further action should be taken to more effectively handle this problem.

5 Acknowledgements

The development of TableMaker was supported in part by National Science Foundation grants DBI-0321600 and DBI-0606909 to V.B. and C.L.

6 References