Astronomical Data Analysis Software and Systems VI
ASP Conference Series, Vol. 125, 1997
Gareth Hunt and H. E. Payne, eds.

QDB: An IDL-Based Interface to LASCO Databases

A. E. Esfandiari, S. E. Paswaters, D. Wang

Interferometrics, Inc.

R. A. Howard

Naval Research Laboratory

Abstract. QDB is a collection of IDL and C routines that provides a query interface to the Large Angle Spectrometric Coronograph (*LASCO*) databases maintained under the Sybase database management system. IDL widgets are used extensively to display the databases, tables, columns, and on-line help. This is a fully automated process—no code modification is required to reflect database changes such as adding/dropping databases, tables, or columns. Standard Query Language (SQL) is used to build a query based on the user selection. This query is then passed via Remote SHell (`rsh`) to two C routines that access the Sybase Open Client Database library to execute the query. The result is returned in an IDL structure. Another set of IDL routines optionally displays or manipulates the data in this structure.

1. Introduction

LASCO is one of the first *LASCO* instrument databases to be defined and populated with data. It is relatively change-free now but, while in the test mode, it went through many structural changes. These changes illustrated a need for a general and flexible interface that could be used with any database, regardless of its structure, size, and type of data. We needed a robust interface that did not require editing if, for example, a new table was added, or a column name was changed. This led to the creation of QDB, which interacts with a database without any advance knowledge about its structure or even its existence. QDB is a user friendly software package with much on-line help. It is currently used remotely to build and execute a LASCO SQL query, and then capture the returned data. This paper is devoted to explaining the inner workings of QDB as it performs this task.

2. QDB Details

2.1. Building a Query

QDB may be called with or without arguments. If the user knows SQL and has a query ready, he can simply pass the database name and the query as arguments to QDB, bypassing all the widgets. Otherwise, if no argument is provided, the following events take place. Upon invocation, QDB sends a query to the RDBMS

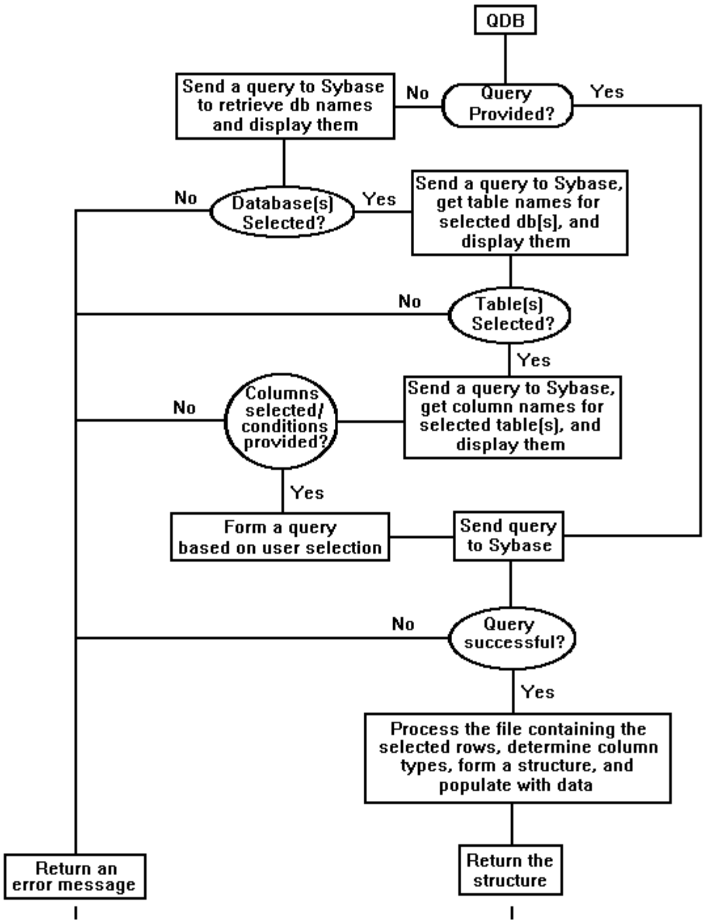

Figure 1. Flow diagram of QDB.

requesting all the existing database names, if any. If successful, it uses the IDL widgets to display them as toggle buttons. The user then clicks on database name(s) to select one or more databases. QDB then sends another query to the RDBMS requesting the table names within the selected database(s). If successful, it uses the IDL widgets again to display the table names using toggle buttons. The user now selects the tables he needs from the displayed list. QDB sends yet another query to the RDBMS, requesting the column names within each selected table, and displaying them using a combination of buttons that allow selection of any column (field) and providing areas for entering optional query conditions. When satisfied with the selections, the user exits the menu. QDB keeps track of all of the user selections, uses them to build a SQL query, and submits it to the RDBMS. At each step of this process, user can seek help using the help buttons, or simply exit the program. Figure 1 shows the flow diagram for QDB.

Because of QDB's dynamic interaction with the RDBMS, it always sees the latest database. Therefore, if a new column is added to a table while QDB is running and is displaying the table names, that new column appears in the columns list when QDB reaches that point. This flexibility makes QDB a low maintenance utility. In fact, only a one-time small code change is required to set conditions for joins between newly added tables, if such joins are desired.

2.2. Interface with RDBMS

QDB itself resides on the client machine but it calls two C routines that access the Sybase Open Client Database library to execute the query and they, along with the RDBMS, reside on the server machine. The query is passed to these routines using UNIX's Remote SHell (`rsh`) service. Besides adding the flexibility of using QDB remotely, `rsh` provides access security since the host machine must know the client login information (user and machine name from which QDB is started) beforehand.

2.3. The Output

The result of the query, along with each column's name, length, and data type, is captured in a text file on the client. QDB parses this file and builds a structure with field names and types corresponding to the selected column names and data types, and populates it with the data. This IDL structure is then returned to the user. Another set of IDL routines is provided to display or manipulate the data in this structure.

2.4. About The Code

This interface consists of about 1200 lines of IDL code and another 550 lines of C code. It is currently setup for use with UNIX and Sybase systems but it can be modified for use with other operating systems and/or relational database management systems. To use another operating system, `rsh` must be replaced with other means of communication with the remote C routines. To use another RDBMS, the C routines must be modified to make calls to the library routines specific to that RDBMS.

3. QDB and LASCO Database

LASCO is one of the *LASCO* instrument databases whose basic structure has been relatively change-free but has been regularly updated with new data. Composed of JPEG browse images and tabular data about these images, it is growing rapidly. Currently, it uses 450 MB of disk space to archive over 80,000 compressed browse images and related information. The database is organized into tables containing compressed images, image names, image header information, image types, image processing steps, image parent information, image history, etc. QDB is currently used to query this database. Figure 2 is a sample display where two tables, *img-browse* and *img-leb-hdr*, have been selected and displayed, and user selection is in progress. In this figure, *filename*, *browse-img*, *data-obs*, *camera*, *filter*, and *polar* fields are toggled on for selection and a time range using the *data-obs* field is entered.

QDB is capable of joining various tables within the LASCO database to produce the desired output, but, more importantly, it is also capable of joining tables from different databases. This is an important feature because we will soon have other databases in production, such as housekeeping, observing,

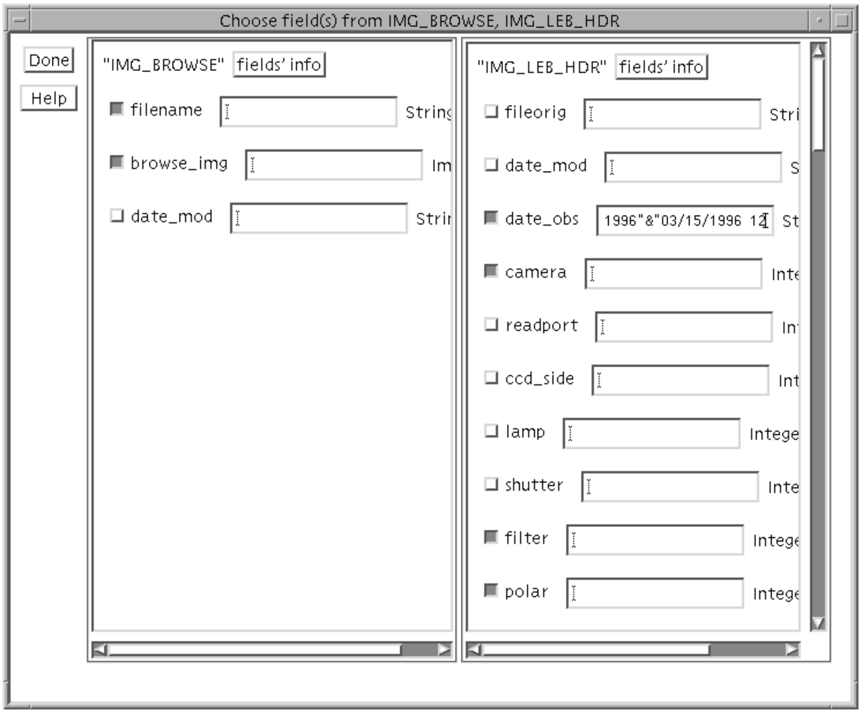

Figure 2. A sample display. Two LASCO tables selected for user interaction.

calibration, user access, and processing tables which may require joins. As mentioned earlier, the dynamic nature of QDB allows for the addition and linkage of these databases with only a minimal amount of additional effort.

4. Conclusion

Because of its dynamic nature and flexibility, QDB can be used remotely to access any database. A user with prior permission, for instance, may run the QDB from his machine over the network, access the LASCO database on our host, and capture the result on his client. Moreover, QDB users need not know SQL in order to use it. All that is needed is the knowledge of which database and tables contain the desired data. In most cases, the database and table names, themselves, convey this information. Furthermore, since result of a QDB query is captured in an IDL structure, it is available and ideal for programmatic use such as displaying, graphing, and data reduction. These features of QDB have allowed us to query the LASCO database frequently and efficiently. We also use QDB to test other databases that are currently under development. In these cases, the only overhead is a one-time minimal code change to handle new joins.