J2.5 ***NQuery: a network-enabled data-based query tool for multi-disciplinary earth-science datasets***

John R. Osborne[1], Kevin T. McHugh[2,3], and Donald W. Denbo[2]
[1]OceanAtlas Software, Vashon, WA
[2]Joint Institute for the Study of the Atmosphere and Ocean, University of Washington, Seattle
[3]Coastal Data Information Program, Scripps Institution of Oceanography, La Jolla, CA

1. INTRODUCTION

NQuery is a Java-based, network-enabled data-based query tool that loads pertinent subsets of multi-disciplinary, earth-science datasets into a temporary, on-the-fly relational database, performs local calculations, and then allows a scientist to construct sophisticated SQL queries. What distinguishes NQuery from other data selection tools is the ability to find a subset of a dataset from the values of measured parameters rather than just the spatial domain. NQuery currently works only with oceanographic profile data but can be extended to time-series data (and other types of data) in future revisions.

NQuery computes summary statistics (e.g., average value, depth of maximum value, and depth of minimum value) for a profile from the observed parameters and from user-specified calculations (e.g., mixed-layer depth, apparent oxygen utilization, and interpolation of a measured parameter to a standard level). A simple two-step process takes the user from pre-selected data (e.g., from local data collections, or distributed data collections available through Dapper/OPeNDAP) to a set of data files selected from the computed summary statistics. First, the user determines what observed and computed variables will be used to construct the database, and therefore are available for subsequent queries. This selection is accomplished via a simple graphical interface that lists all available variables. Additional dialogs allow selection and configuration of a large set of computed variables. A single click then begins the process of ingesting the data files, computing summary statistics for both the observed values and user-specified calculated values, and building and populating a MySQL database. Second, after the database has been populated with the requested data, the user can create a SQL query using a second graphical user interface.

The user can build simple to fairly complex queries by using either the graphical interface, or entering an SQL statement by hand. Once a satisfactory query is constructed, it is executed by the database, resulting in a list of files that satisfies the query. The scientist can then use these files in other research tools by exporting a "pointer file" that contains the locations of the actual data files.

* *Corresponding author address:* John Osborne, NOAA/PMEL/OD, Route: R/PMEL, 7600 Sand Point Way NE, Seattle, WA 98115-6349 (john.osborne@noaa.gov)

2. NQUERY ARCHITECTURE

The major components of NQuery are shown in Figure 1. NQuery is a Java application with built-in components to communicate with MySQL database servers (JDBC) and Dapper-enabled databases (Sirott, 2004). The JDBC drivers handle communications between MySQL database servers that are installed on your desktop personal computer as well as those accessible over a network. In each case, NQuery's internal database-access API communicates with the JDBC driver that in turn communicates with a specified MySQL database server (mysqld). NQuery can ingest data from Dapper databases through the Dapper Wizard and from files on your local system via an EPIC XML pointer file. NQuery also creates and reads it's own "database documents" — an XML file that contains information to reopen and query existing NQuery on-the-fly database. Currently NQuery produces two output files, database documents (described above) and EPIC XML pointer files for query results from an NQuery database. Currently, the only desktop application that can ingest XML pointer files from NQuery is Java OceanAtlas 4.0 (http://odf.ucsd.edu/joa).

3. USING NQUERY

Before using NQuery, it must be configured to communicate with a local or network-accessible MySQL database server (Figure 2). NQuery requires the URL of a database server and a password and username. Other preferences include which built-in calculations to perform on ingested profile data (Figure 3) and how NQuery handles variables names and variable substitutions (Figure 4) from data files with different naming conventions.

There are two ways to ingest data and create NQuery's on-the-fly databases. For data that resides on your local machine, you first construct a "pointer" file that describes the data's geospatial and temporal domain, measured variables, and individual profiles and collections of profiles (filesets). The pointer file also contains attributes that describe the location of individual data files (pathname) on your local machine. The pointer file is an XML file that adheres to conventions established by the EPIC group at PMEL (Appendix 1).

After successfully parsing a pointer file, NQuery presents a variable selection window where users can select which of the measured variables in the dataset to include in the on-the-fly database (Figure 5). Using additional interfaces, the user can also chose from a large

set of calculated variables to include in the final database (Figures 7-9).

After choosing variables and calculations, the user specifies the name of the database to create and an optional description (Figure 10). This information is written to an NQuery database document that resides on your local machine. NQuery then builds the relational database given the variables the user has specified. By locating the actual data files either through the path stored in the pointer file or through the Dapper interface, reading the actual data files, performing any user calculations on individual profiles, computing summary statistics, and finally populating the database with the results (Figure 11).

To use network data, the user begins by invoking the Dapper Wizard. The Dapper Wizard is a three-panel user interface similar in function to many software installers that requires the user to perform actions in a set order.

The first Dapper panels allows the user to specify the URL of a Dapper server (Figure 11). After a Dapper URL has been entered, the tree view in the middle panel shows collections of data sets available on the chosen server. Clicking a folder icon in the list shows the actual file sets in a collection (Figure 11 shows the interface after a particular collection has been opened). The user then selects a file set and clicks the "Select Domain" tab to show the second step of the wizard (Figure 12).

The Select Domain tasks displays the geospatial and temporal domain of the selected file set and shows the number of distinct files available. Controls allow selecting a subset domain to reduce the amount of data returned by the servers (historical data collections can be very large.) You can also specify a sub-domain to filter the data to meet your research needs. After you have specified a domain, you click the "Select Stations" tab to display the wizard's third panel for selecting the actual files to return from the server.

The third panel is based upon a tool called NdEdit (Osborne and Denbo, 2002) that combines tools for filtering in space and time along with a set of selection tools for precisely specifying which files to return from the database server. After you have selected the files to use (shown in blue in Figure 13), you click the "Select" button at the bottom of the wizard window show the variables measured in the selected files in the NQuery variable selections window (Figure 5).

From this point using NQuery is identical to the steps described above for selecting variables, specifying preferred built-in calculations, and defining any user-defined calculations. In the case of Dapper data, NQuery retrieves the actual data from the server in OPeNDAP format. Using code created by PMEL's Donald Denbo NQuery converts the OPeNDAP structures into a netCDF representation before reading and processing. Figure 14 shows the database document window opened when creating a new database. The upper log panel shows the progress of the database construction and data ingest processes. Figure 15 shows the "About" windows for a database just created by NQuery.

Once NQuery has built and populated a database of summary statistics, NQuery can be used to compose SQL query statements in a simple point-and-click user interface to use the database to identify profiles relevant to specific research needs. The query interface is the middle panel of a database document window. A simple example of the query interface is shown in Figure 16. Here the user wants to select all profiles where the mean measured temperature is between 10 and 15 degrees. Note: as you use the point-and-click query builder, NQuery builds and displays the actual SQL syntax in the lower panel. The results of a query are shown in Figure 17. To save the selected files to a new XML pointer file for further analysis, click the "Save" button.

The query interface can be used to construct sophisticated SQL queries but has limitations (e.g., no way to group criteria with parentheses). Figure 18 shows a more complicated query with three criteria. To specify an arbitrary SQL query, click the "DB Command" button to enter any valid MySQL command and send it to the database server.

4. SYSTEM REQUIREMENTS

NQuery is compatible with all major desktop operating systems including Windows, Mac OS X, and Linux. NQuery requires version 1.4.2 or greater of the Java virtual machine. NQuery has been tested with version 4.1 of MySQL. An Internet connection is required for access to network data through Dapper and to create databases on networked MySQL database servers.

5. INSTALLING NQUERY

The NQuery application is installed with a standard double-clickable installer for Windows, Linux, and generic UNIX. The Mac OS X version of NQuery is provided as a mountable disk image. On Mac OS X, installation is by dragging the NQuery folder to the Mac's hard disk. Installation of MySQL is beyond the scope of this paper but is described at: http://dev.mysql.com/doc/mysql/en/Installing.html

6. LOOKING TO THE FUTURE

Currently, NQuery only works with profile data. To accommodate time series data, NQuery will need to define what summary statistics are computed and any user-defined calculations. The query interface will also be enhanced to be able to create queries with grouped criteria.

7. REFERENCES

Osborne, J.R. and D.W. Denbo, 2002: NdEdit: Interactive Data Selection Tool. 18th Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology, AMS, 13–17 January 2002, Orlando, FL

Sirrott, J, D.W. Denbo, and W. Zhu, 2004: Dapper: an OPeNDAP server for in-situ data. 20th Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology, AMS, 10–15 January 2004, Seattle, WA.

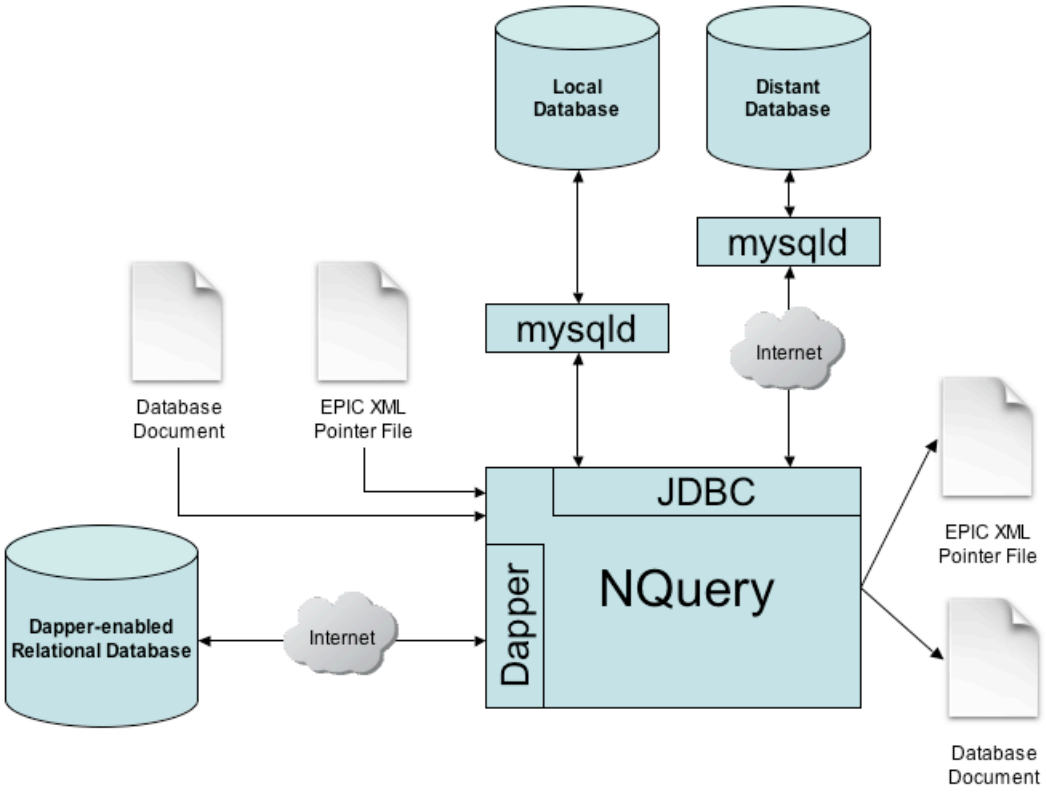

Figure 1. NQuery Architecture

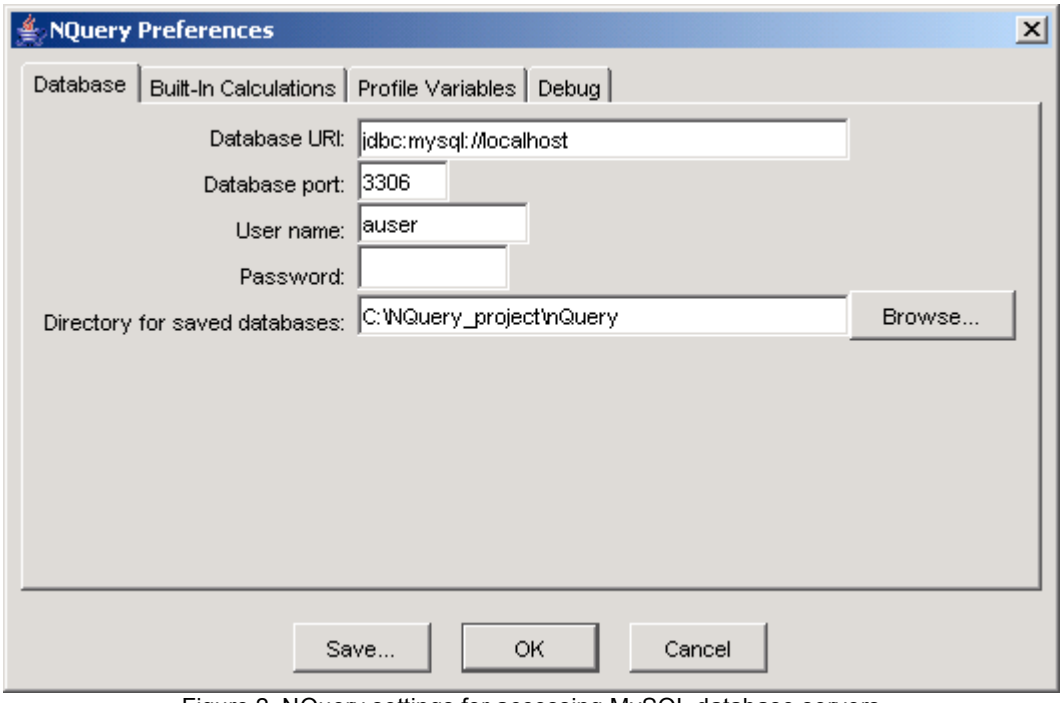

Figure 2. NQuery settings for accessing MySQL database servers

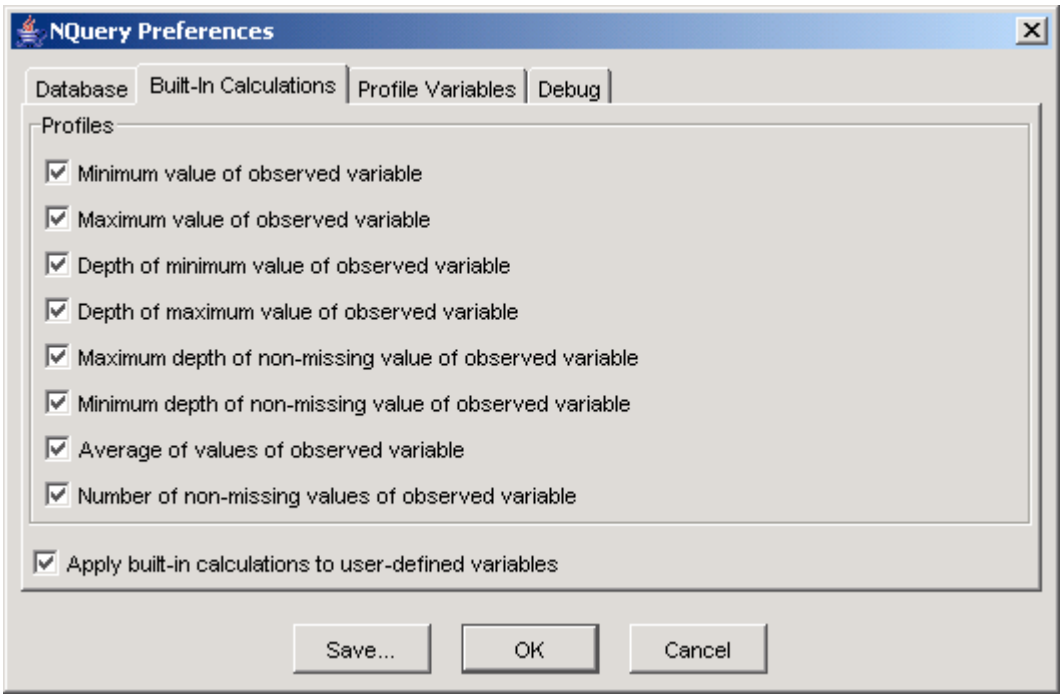

Figure 3. NQuery preferences for selecting the built-in profile data calculations

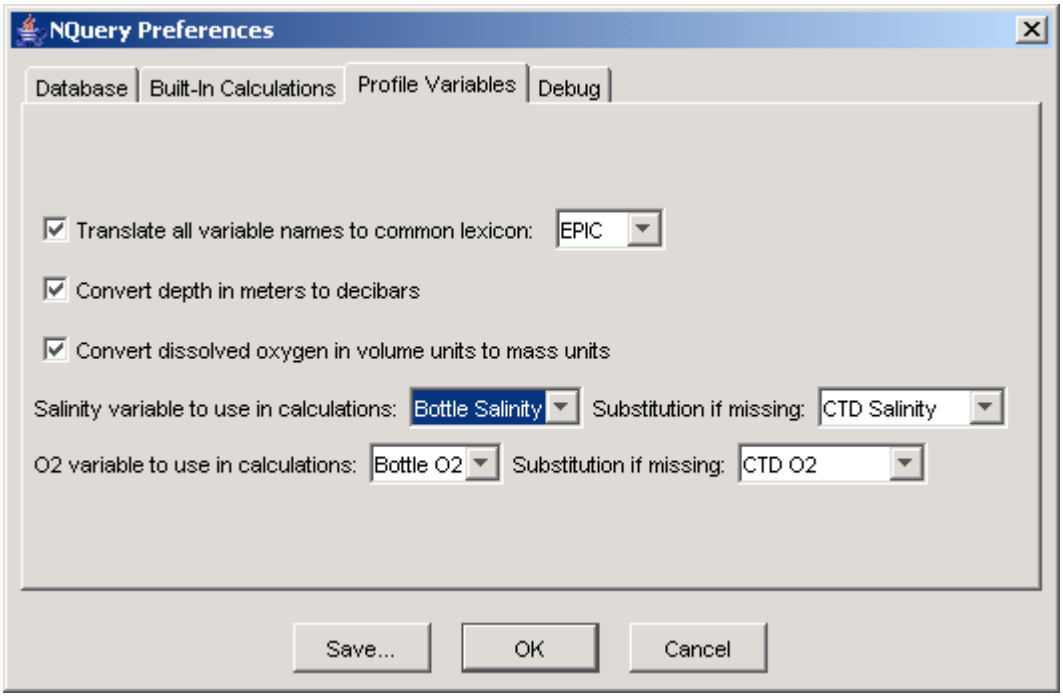

Figure 4. NQuery preferences for handling variable names and units from different lexicons and settings for how to handle missing values for variables required computing new variables (e.g., theta, density, NO, and PO)

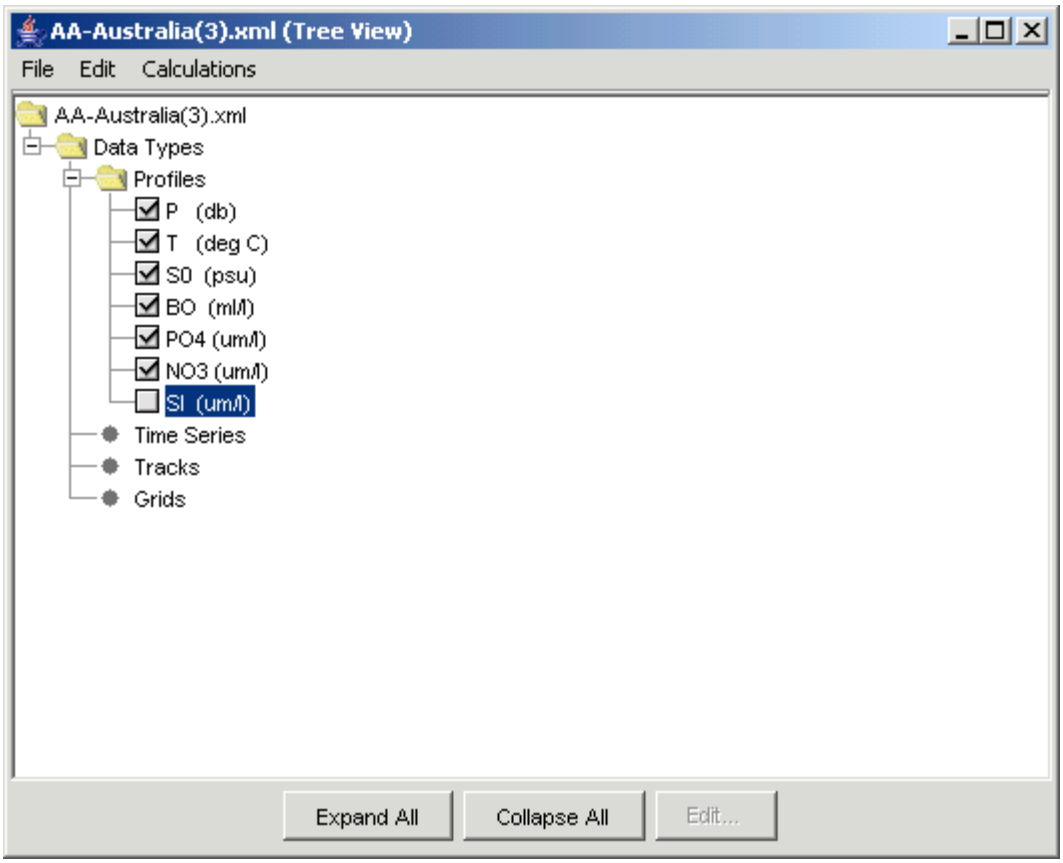

Figure 5. NQuery variable selection window

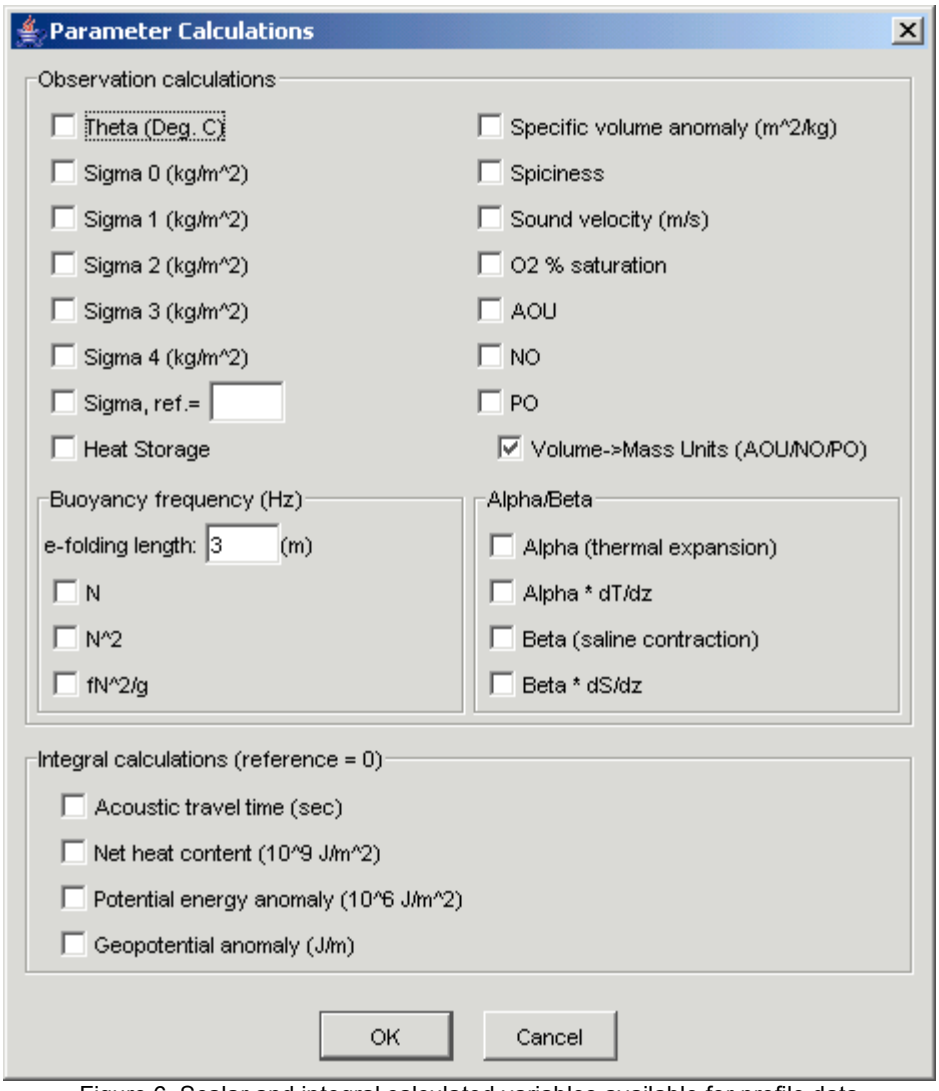

Figure 6. Scalar and integral calculated variables available for profile data

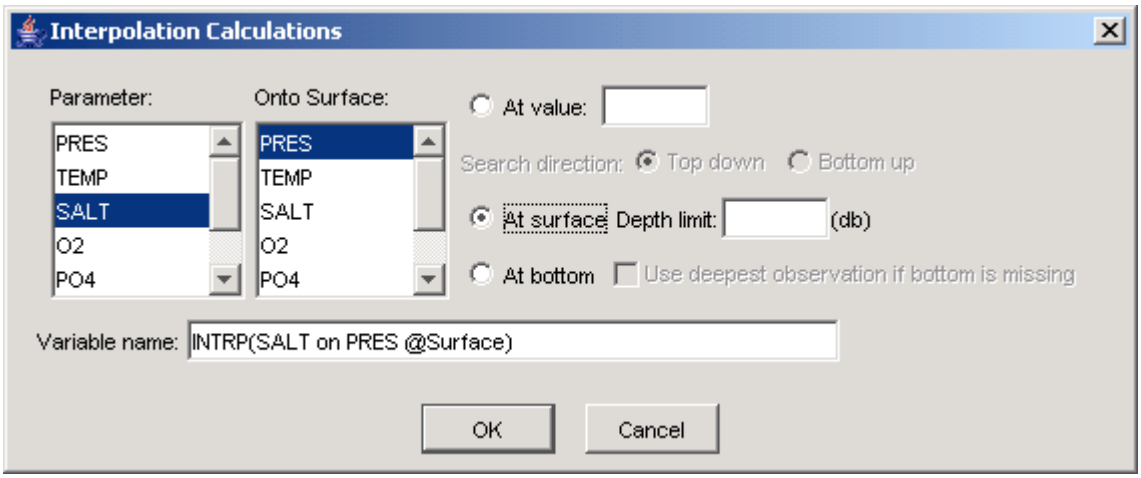

Figure 7. Settings for interpolating a measured variable onto a value of another variable

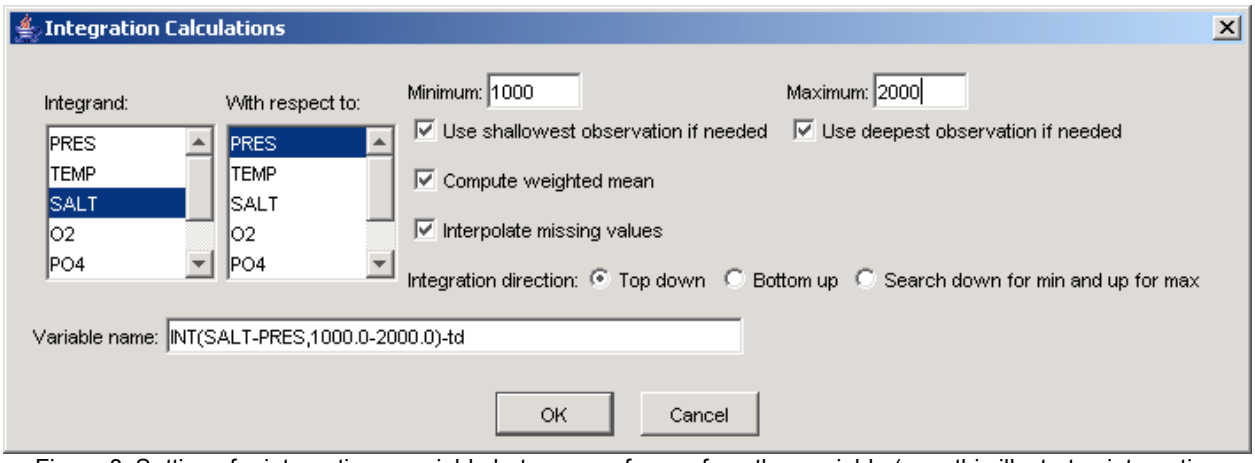

Figure 8. Settings for integrating a variable between surfaces of another variable (e.g., this illustrates integrating salinity between the 1000 and 2000 decibar pressure surfaces)

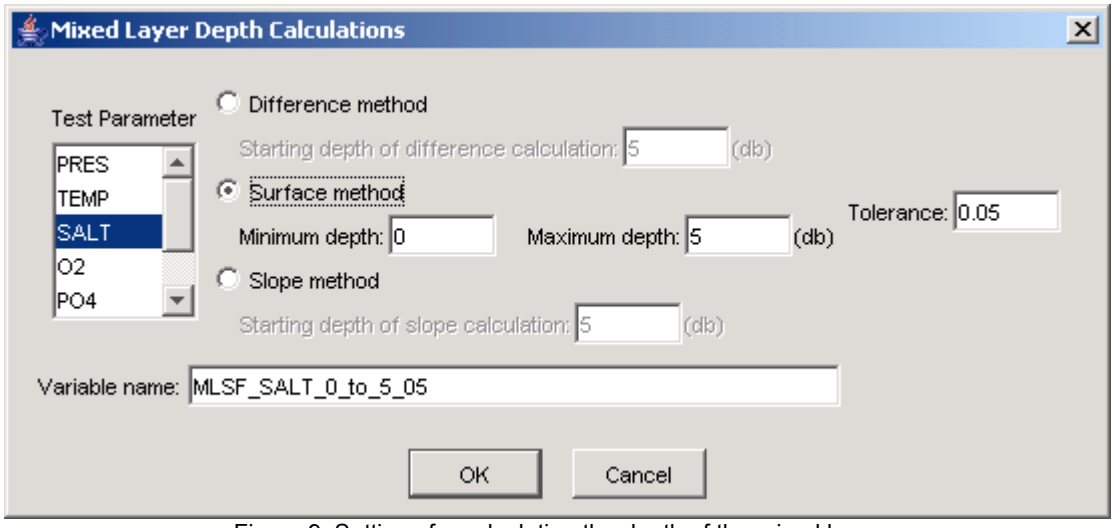

Figure 9. Settings for calculating the depth of the mixed layer

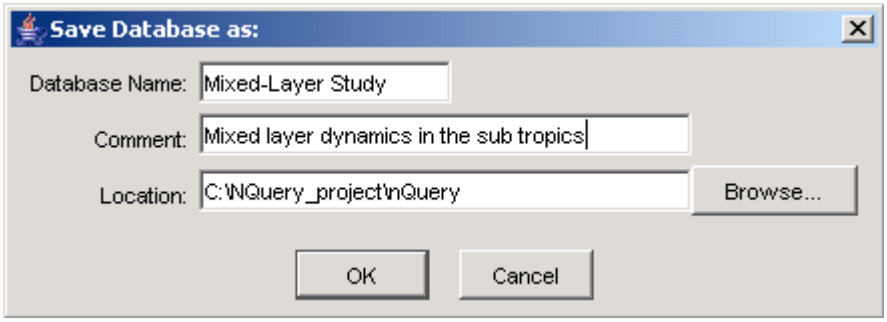

Figure 10. Dialog allows naming that on-the-fly database and attaching an optional comment. A database document file will be created on disk with this name and at the location specified.

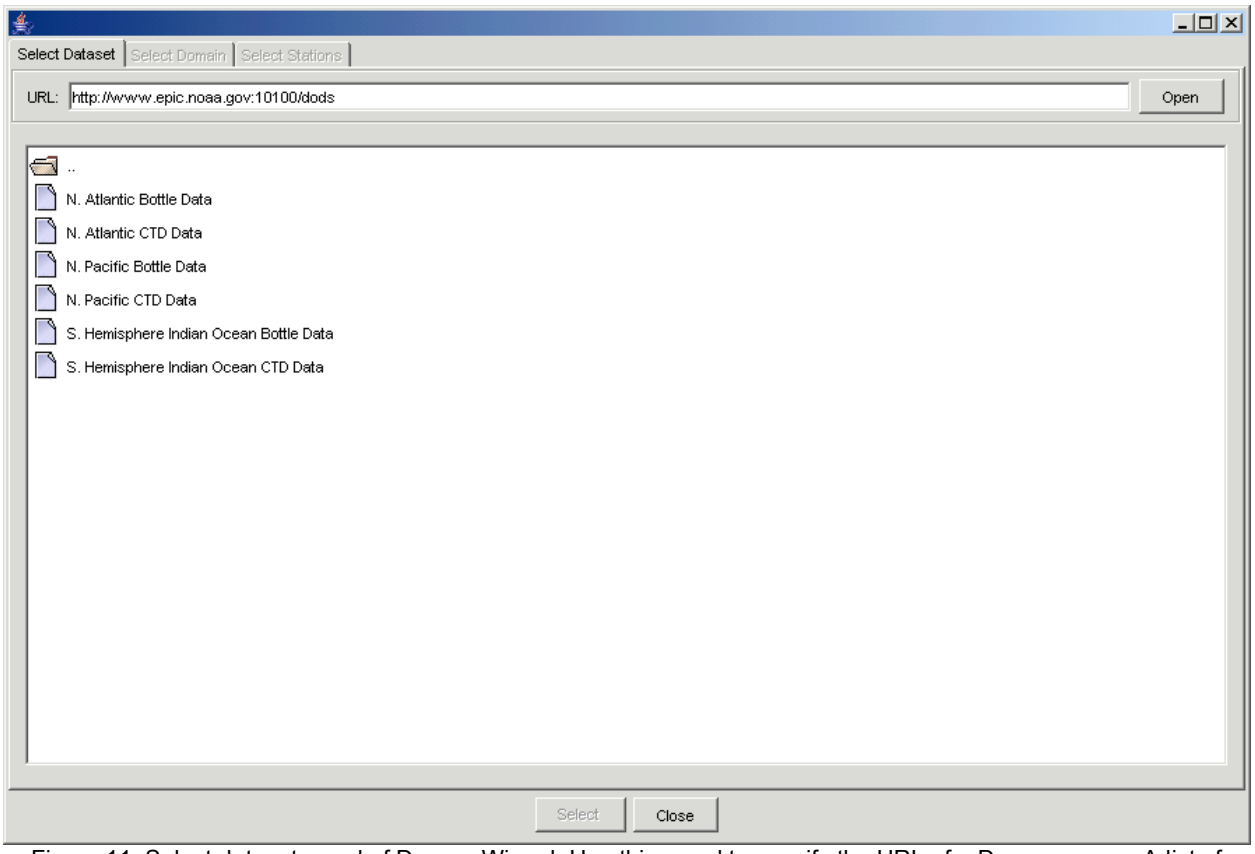

Figure 11. Select dataset panel of Dapper Wizard. Use this panel to specify the URL of a Dapper server. A list of datasets available on the selected server is shown in the middle panel.

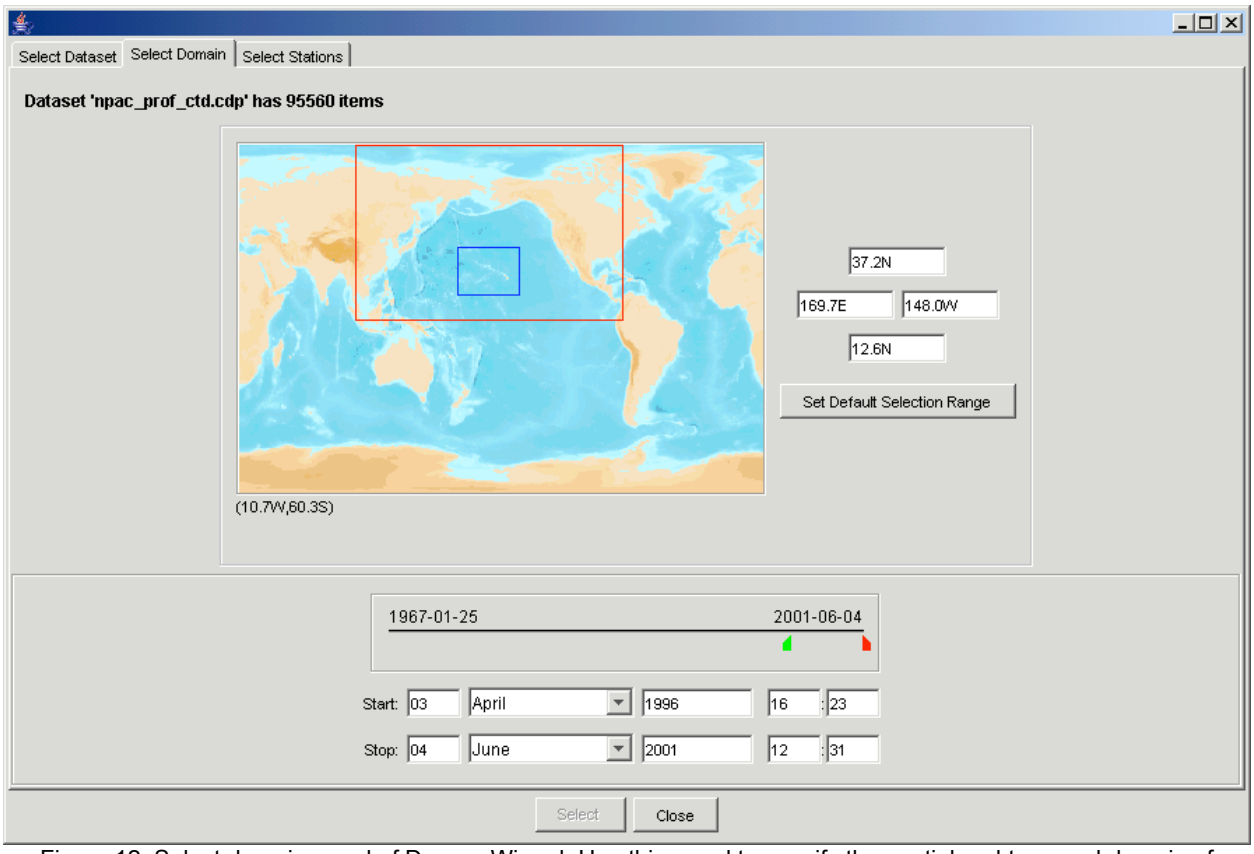

Figure 12. Select domain panel of Dapper Wizard. Use this panel to specify the spatial and temporal domain of profiles to return from the Dapper server.

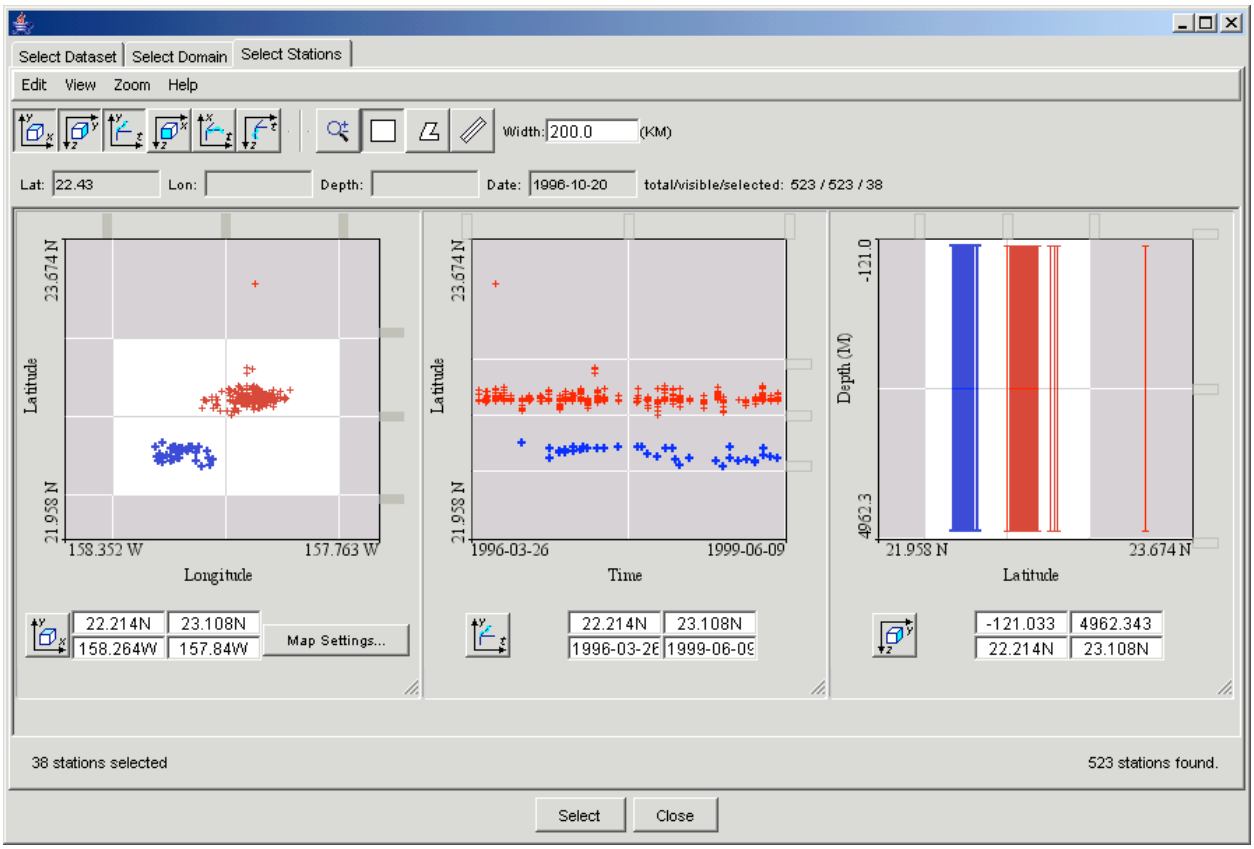

Figure 13. Select stations panel of Dapper Wizard uses NdEdit (Osborne and Denbo, 2002) to filter results from Dapper and select those profiles (shown in blue in this figure) to use to create NQuery database.

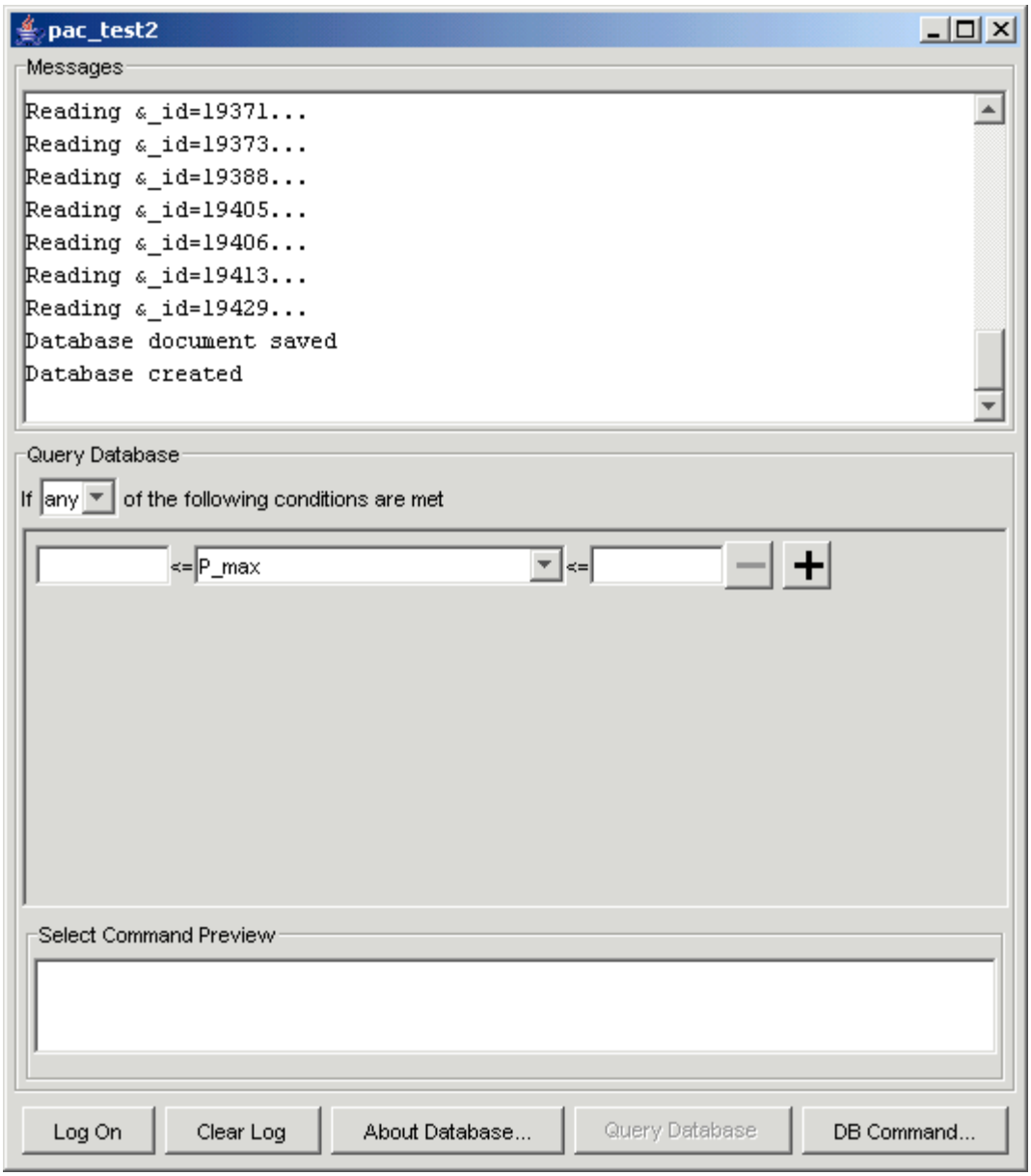

Figure 14. NQuery database document window. This window shows progress of database creation (top panel) for new database and allows construction of SQL queries (see Figure 15) for both new databases and saved databases.

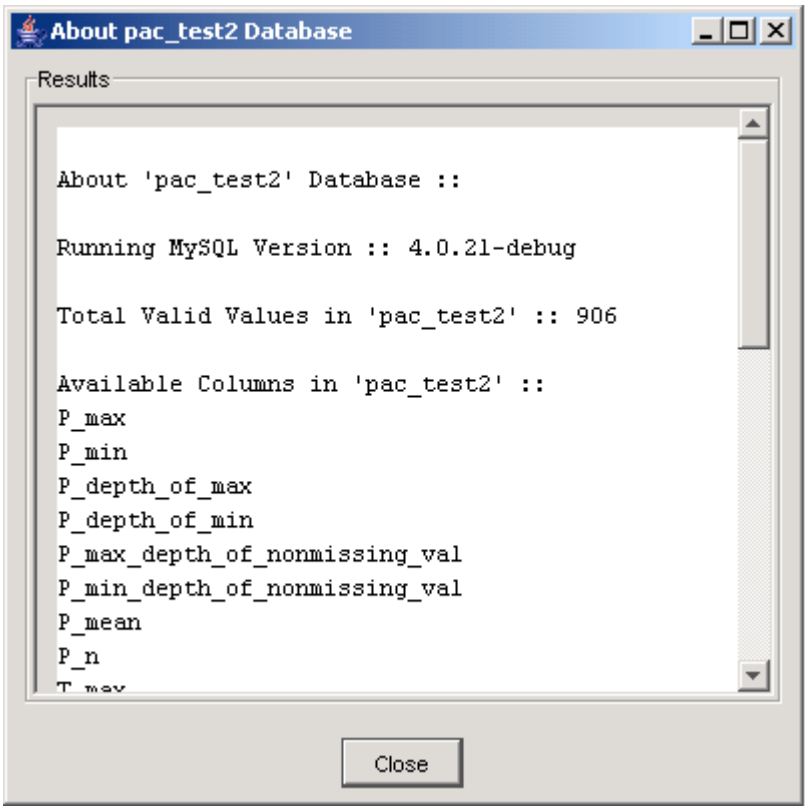

Figure 15. About Database window shows summary information for current database.

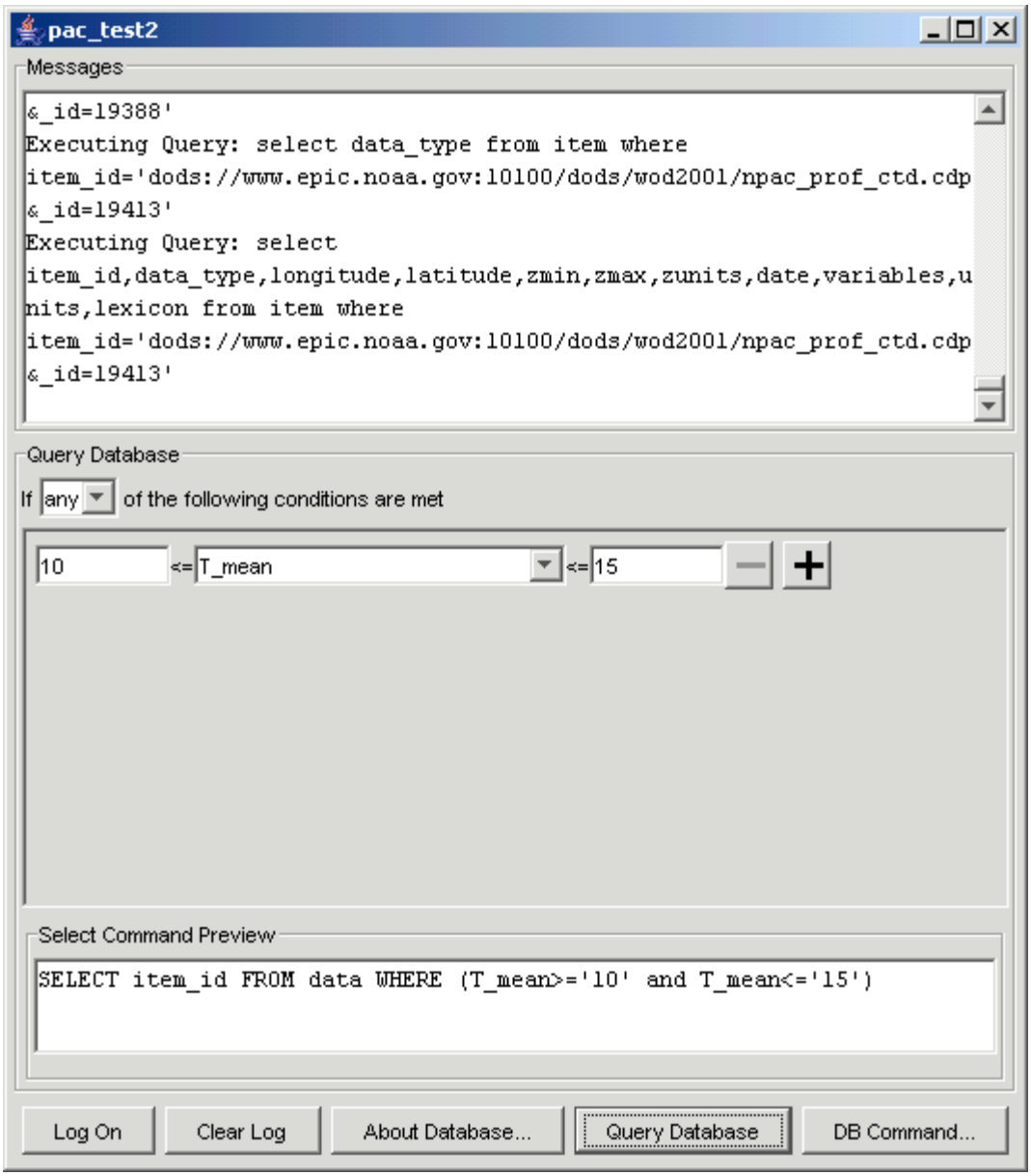

Figure 16. Database document window showing a simple query in the middle panel, the actual SQL command in lower panel, and the results of the query in the top, message panel.

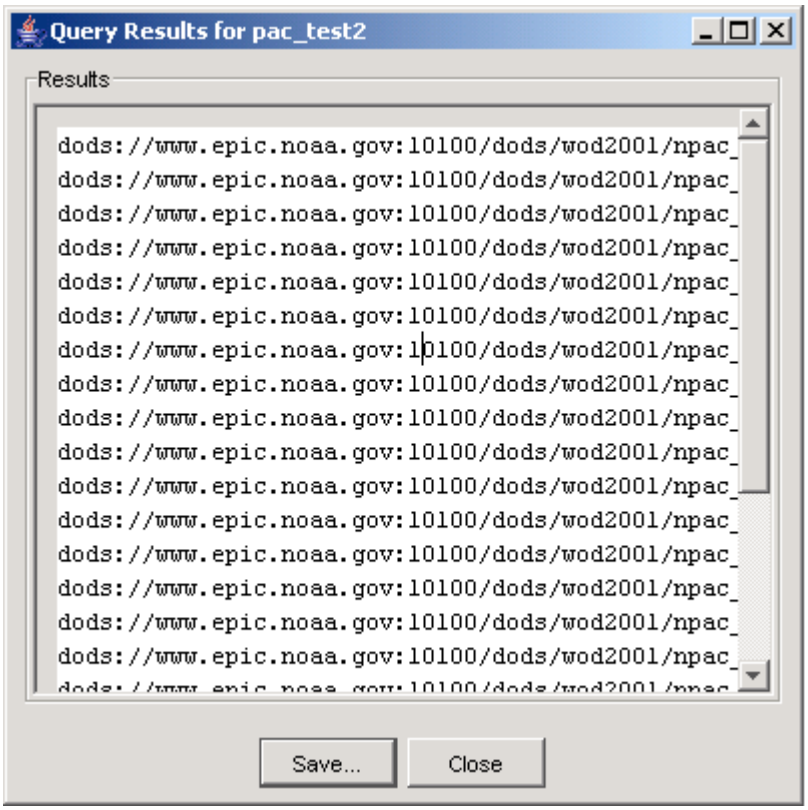

Figure 17. Query results window. Click the Save button to save query results as XML pointer file.

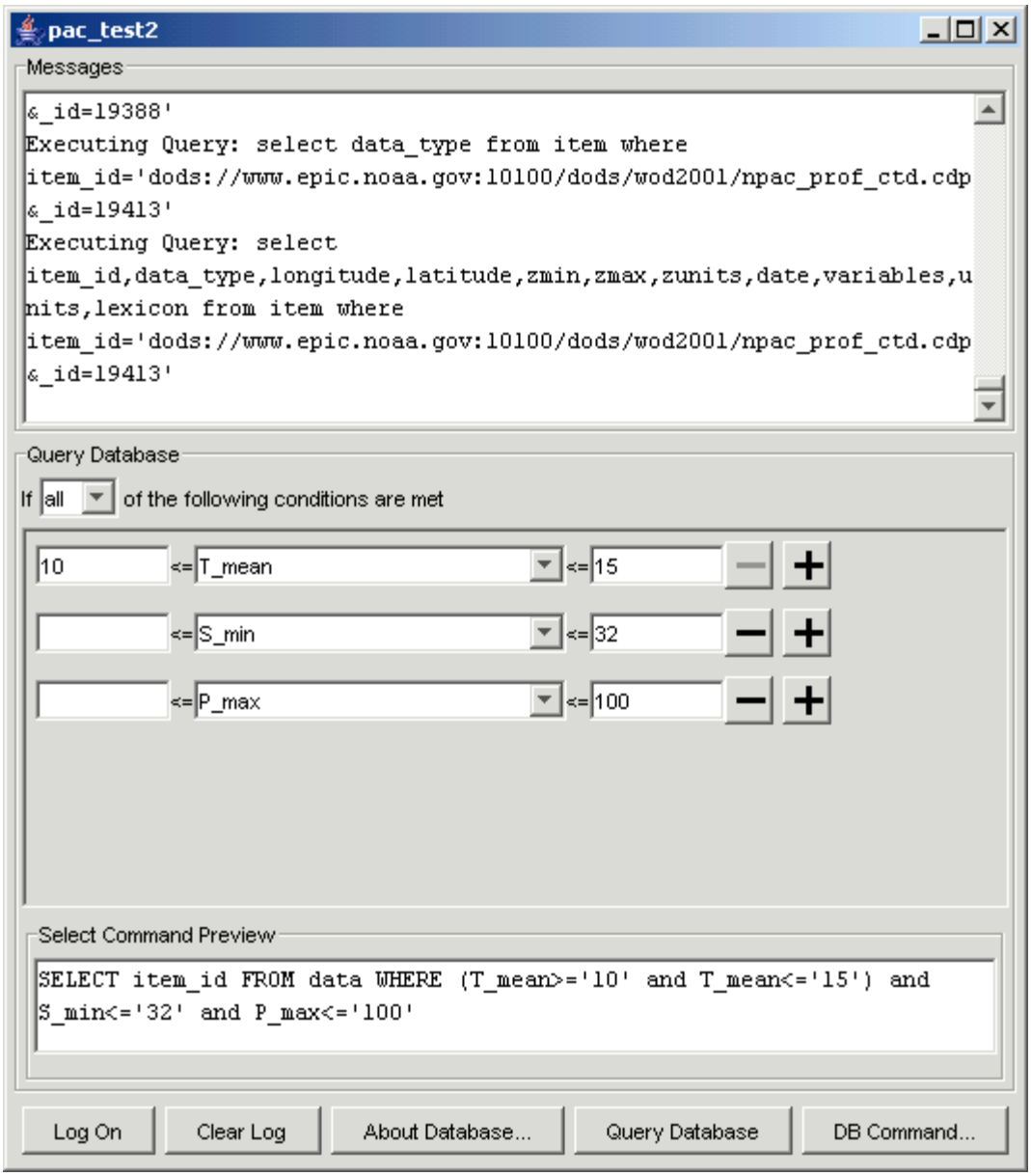

Figure 18. Database document that illustrate more complex query constructed with the query interface in middle panel.

APPENDIX 1: DTD SPECIFICATION FOR EPIC XML POINTER FILES

```
<!--DTD for EPIC Profile and Time-Series Data -->
<!-- ? =  optional, not repeatable -->
<!-- + =   required and repeatable -->
<!-- * =  optional and repeatable -->

<!ELEMENT epicxml(domain, varlist?, fileset+, attribute*, comment*)>
<!ATTLIST epicxml version CDATA #REQUIRED
                  type (profile | time-series | grid | other) #REQUIRED
                  URI CDATA #IMPLIED
                  lexicon CDATA #IMPLIED>

<!-- domain for epicxml must consist of location="north", "south", "east",
"west", "top", "bottom", "start",  and "end" for latitude, longitude, vertical,
and time, respectively -->
<!ELEMENT domain(latitude+, longitude+, vertical+, (time | date)+, attribute*,
comment*)>

<!ELEMENT deltat (#PCDATA)>
<!ATTLIST deltat units CDATA #IMPLIED "minutes">     <!-- acceptable units are
seconds, minutes, hours, days, unacceptable units include months, years -->

<!ELEMENT time (#PCDATA)>
<!ATTLIST time location (start | end | point) #REQUIRED "point"
              units CDATA #IMPLIED >

<!ELEMENT date EMPTY>
<!ATTLIST date location (start | end | point) #REQUIRED "point"
               year CDATA #REQUIRED
               month CDATA #REQUIRED
               day CDATA #REQUIRED
               hour CDATA #IMPLIED
               min CDATA #IMPLIED
               secs CDATA #IMPLIED>

<!ELEMENT latitude (#PCDATA)>
<!ATTLIST latitude location (north | south | start | end | point) #REQUIRED
"point"
                   units    (degrees_north    |    degrees_south)    #IMPLIED
"degrees_north">

<!ELEMENT longitude (#PCDATA)>
<!ATTLIST longitude location (east | west | start | end | point) #REQUIRED
"point"
                   units (degrees_east | degrees_west) #IMPLIED "degrees_east">

<!ELEMENT vertical (#PCDATA)>
<!ATTLIST vertical location (top | bottom | start | end | point) #REQUIRED
"point"
                  units CDATA #REQUIRED
                  positive CDATA "down">

<!ELEMENT varlist ((variable | variableref)+, attribute*, comment*)>
<!ATTLIST varlist lexicon CDATA #IMPLIED>

<!ELEMENT variable (attribute*, comment*)>
```

```
<!ATTLIST variable name ID #REQUIRED
                   units CDATA #REQUIRED
                   description CDATA #IMPLIED
                   lexicon CDATA #IMPLIED>

<!ELEMENT variableref EMPTY>
<!ATTLIST variableref refname IDREF #REQUIRED>

<!ELEMENT fileset (varlist?, station*, grid*, track*, attribute*, comment*)>
<!ATTLIST fileset id CDATA #REQUIRED
                  URI CDATA #IMPLIED>

<!-- time should have location="start", "end" for time series, "point" for
profile,
vertical should have location="top", "bottom" for profile, "point" for time
series,
latitude and longitude should have location="point" -->
<!ELEMENT station (varlist?, (time | date)+, latitude, longitude, vertical+,
deltat?, stationvalues*, attribute*, comment*)>
<!ATTLIST station id CDATA #REQUIRED
                  cast CDATA #IMPLIED
                  URI CDATA #IMPLIED
                  bottom CDATA #IMPLIED
                  reference CDATA #REQUIRED>

<!-- domain for grid must consist of location="north", "south", "east", "west",
"top", "bottom", "start",  and "end" for latitude, longitude, vertical, and time,
respectively -->
<!ELEMENT grid (varlist?, domain, attribute*, comment*)>
<!ATTLIST grid id CDATA #REQUIRED
URI CDATA #IMPLIED
reference CDATA #REQUIRED>

<!-- domain for track must consits of location="start" and location="end" for the
latitude, longitude, vertical, and time elements -->
<!ELEMENT track (varlist?, domain, attribute*, comment*)>
<!ATTLIST track id CDATA #REQUIRED
URI CDATA #IMPLIED
reference CDATA #REQUIRED>

<!ELEMENT stationvalue (attribute*, comment*)>
<!ATTLIST stationvalue cast CDATA #REQUIRED
                 name CDATA #REQUIRED
                 units CDATA #REQUIRED
                 method CDATA #IMPLIED
                 lexicon CDATA #IMPLIED
                 value CDATA #REQUIRED>

<!ELEMENT attribute EMPTY>
<!ATTLIST attribute name CDATA #REQUIRED
                    value CDATA #REQUIRED>

<!ELEMENT comment (#PCDATA)>
```