The Manual for the Query tool- Part of pg_browse

The Query tool is a tool that is built by the pg_browse program. It is built for a specific class, and has the class, database, and postgres host built into the code. The code allows a user to perform queries on a postgres class without having to learn Postquel. Typical operations that can be performed are look ups, entering, and deleting from the database. In addition to these actions you can perform actions on the data you retrieve, including actions such as displaying or animating in IDL, Copy large objects to local disk, edit or remove a single instance from a class.

Query Format:

The queries are formed by entering information into the field or fields you wish to query on (see Figure 1- Part 4) . The query tool creates a box for each field in the class, and you can enter in information into the box to select the instances you want.

Examples:

1.) Suppose you have a class that has Julian Days, and the field is called jday. To look up what data you have for day 310, Just put a 310 in the jday field and hit return, this will return all instances with 310 as the jday.

2.) Suppose you are only interested in a certain hour, by leaving the 310 in the jday field and entering an 18 in the hour field you will retrieve all instances that have 310 as the day, and 18 as the hour.

3.) If you want to do ranges and find all jdays>300 and <360 you can put two numbers in the jday field separated by a space. This will retrieve all jdays meeting this condition 300>jday>360. This will only work with numeric fields not text.

4.) Text lookups do a Case-Sensitive substring search. Entering Smit into a field of author names will retrieve any entries that contain Smit, i.e. Smith. Being Case-Sensitive if you enter smith you will NOT retrieve Smith.

The query formation is used in queries to the database and deletions from it. All the instances matching the query will be displayed in an output window below the query tool. Deletions have a further limitation in that will be described below.

Hitting the Tab key while in the entry fields will bring you to the next entry field. You can place the mouse on the field you want, and that will focus keyboard entry to that field. Mouse button 3 will paste from an Xbuffer, and mouse button 2 will scroll if the entry field is longer than the current screen.

Looking Up From Database:

When you have formed the query you wish to look up, just hit return to do the look up, or hit the look up button, or select the look up from the file menu. As look up is the most common function, return is binded

to it, to make it simple to do look ups. The Output of the Postquel command will be displayed in a window down below. (See Figure 8)

Deleting From Database:

The Query tool supports two types of deletions. The first type is available via the file menu in the main Query window or the button on the screen. This is the 'Safer' deleting process, to catch typos from entering data. This method has a hardwired safeguard that will not delete any instances over an hour old. It will follow all the rules of the queries above, so it is possible to delete quite a lot, on a new database less than an hour old.

The Second type of deletion is available through the edit_instance action. The edit instance action is in the action menu on the query output window. First you query the data base, then you can select an instance or instances to edit. To select one, just double click on it this will bring you into edit mode. To select more than one just drag the mouse with button one over the instances you wish to edit or delete, and select edit instance from the action menu.

In edit mode it will put the instance you selected back into the query tool. It will also bring up a new window with some important actions in the file menu. If you selected more than one instance it will cycle through all the instances each time you use one of the edit actions. The delete action in the edit mode will allow you to delete any instance no matter how old it is. It will only delete one instance from the database, no query matching is done.

Edit Mode:

Edit mode was described somewhat above in the delete section, but edit mode does more than just delete. After selecting an instance by double clicking in the output window, or multiple instances by selecting then performing the action, you will be placed in edit mode. The delete function allows you to delete any instance, no matter if it is older than 1 hour. The Edit Mode Window is shown in Figure 7.

The replace option allows you to make changes in the entry fields of the query tool and then enter those changes into the postgres database. This will change that instance, it will not add a new instance. If you wish to make a change and enter in a new instance use the Enter into Database command. Replace will allow you to correct errors you might come across in your data. It is easier than doing a delete, then an enter.

The final option in edit mode is cancel. If you are on the last instance or you only edited one, this will get you out of edit mode. If you are editing more than one, this will go to the next one to edit.

Editing mode is a little bit clumsy currently, it pops its menu over the Query tool menu, Actions still have there Query tool bindings, so you could accidentally to a look up while editing.

Entering into Database:

When Entering in data into a database, try to fill in all the fields, no defaults will be given. If you leave any of the fields blank, nothing will be entered for that instance in that field. Fill all the entry fields (Figure 1-Part 4) then hit the Enter into database option. This will add that instance to the postgres class.

Output Window:

The Output Window gets the output of all monitor commands. (See Figure 8). It is a scrollable output, and Mouse button 2 will scroll for quick scanning. Mouse button 1 is for selecting input for actions. You can select more than what is on the screen by starting the select with mouse button 1, then moving to where you want and click Shift mouse button 1. You can also select all of the output, by activating the select all button in the file menu. Also in the file menu is a print output option- This currently just sends the output to lpr, truncating long lines. If your site has a preferred printing method, the person who installs pg_browse can change the printing.

The purpose of the output window is to either just glance at the output, or do an action on the output. To perform an action, you must do 3 things:

1.) Select the instance or instances you wish to perform the action on.

2.) Select the field from the Field menu, to perform the action on. Some actions have default fields.

3.) Then select the action from the Action menu.

Actions:

Actions are ways of using the output from the monitor command. Typical actions include, file retrieval, Displaying in IDL, Animation, or copying one field to a file, or graphing a column of numbers. Actions can be added by each site, and will by dynamically sourced by the program. Some Large object actions defined will allow users to copy their large objects down from heel, and place the objects in either the current directory or $TMPDIR if it is defined. Other actions will copy down a file, display or animate in IDL, then remove the file.

To add your own action, all you need to do is create a directory in the $PGBROWSE/actions directory for your site. Currently there are directories for UCLA, UCSB, and ESRG. Cd to your site directory and make a pgb.tcl file. The following is a pgb.tcl file that is used to copy a column or field to a file.

```
set Name { small_copy }
set Help { small_copy
This function will take a specified filed and copy it to a file
}
proc Action { dir } {
   global field
   source $dir/small_copy
   small_copy $field [selection get]
}
```

The Name is the name you wish to Appear in the action menu. The Help is what will be displayed if the user selects help from the output window on the given action. Finally the Action definition is what will get called when the action get called. The language this is in is TCL. You may call Shell-Scripts or C or Fortran Programs, but you need pgb.tcl to be written in TCL.

There are two thing I pass to the procedure small_copy, $field and [selection get]. $field is a global variable that will contain the number of the field in the selection. [selection get] returns a list containing the current selection or Xbuffer. There is a global variable PGB which will contain all sorts of information that could be useful when defining actions. $dir is passed to Action so that you can place any additional scripts in the directory with the pgb.tcl file. You can then call these scripts or source them as is done above.

I would suggest looking at the actions small_copy to see how to interact with small objects, action_large_copy to see how to deal with large objects, or idl_animate1 to see both how to deal with large objects and communicate to a named pipe IDL. In the idl animate, I open up a pipe to IDL, and write commands to the pipe, to produce idl output with the given large objects of the monitor command.

Editing Postquel:

If the user is familiar with Postquel, enabling this option will bring up the window in Figure 6 when a command is sent to monitor. With this window you can edit the command before sending it, or cancel the command after looking at it. This will enable users to see the postquel commands generated by the query tool, and allow them to change or add in additional postgres qualifiers such as sort or unique.

The 3 buttons underneath the Postquel command allow you to cancel the query, send it with no changes, or send an edited command to the monitor program. This is potentially dangerous as a user could modify a delete command explained above, and remove the qualifier that only allows the objects less than an hour old to be deleted. Please be careful when experimenting, especially when viewing other peoples data.

Miscellaneous Commands:

There are 3 commands in the Misc Menu in the main Query tool window. The first command toggle editing allows the editing of Postgres described above. The other 2 are Retrieve number of instances, and Retrieve Owner.

The Retrieve number of instances will tell you how many instances there are in a class. This is useful to know before you do a retrieve all. In a future release I would like to add a retrieve number of instances that match the query, but I am not there yet.

The Retrieve owner will tell you the postgres user who owns the current class. This is useful if you need information that is not readily available pertaining to the class. You could mail the owner of the database any questions you have, rather than addressing them to everybody.

User Support & Comments:

Pg_browse is now in Release 2.0 please tell us any suggestions for 3.0, comments can be mailed to pg_browse@crseo.ucsb.edu.