Introduction to Running SQL Queries
Against the DAACS PostgreSQL backend using Navicat:

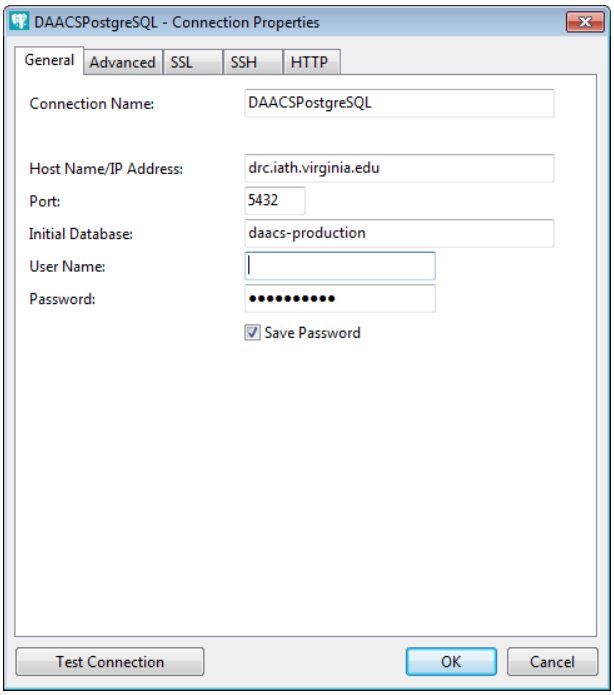
1: The Basics

This document outlines how to use Navicat to write SQL queries and then run them against the DAACS PostgreSQL backend housed at IATH. Navicat, is a database GUI (Graphical User Interface) and administrator tool. Navicat allows you to list the tables and browse their contents in the back end. It also has graphical tools to help write SQL queries.

1. Setup

Navicat in available for download on the web. It comes in different flavors. You want the one for PostgreSQL: http://www.navicat.com/download/navicat-for-postgresql. Navicat offers a 14-day free trial.

Once you have Navicat installed on your machine, you can configure a connection as follows: <double click> the connection icon. This brings the dialog box to the right. Fill in *Connection name*, *IP Address* and *Initial Database* as shown. Then fill in the *User Name (*"drcquery"*)* and *Password* ("!queryacct!") .

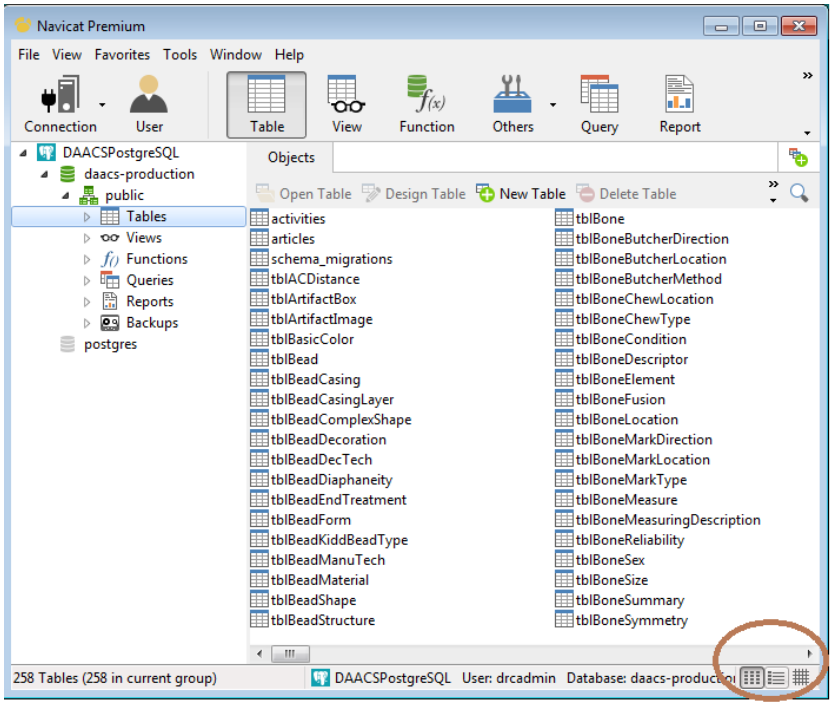


Now hit *Test Connection*, If you see *"Connection Successful",* you are all set. If not, you need to troubleshoot. You can find some tips here: https://help.navicat.com/hc/en-us/articles/217791058-Why-I-cannot-connect-to-my-server- . A likely culprit is the firewall that protects the network on which you are located. To address that, you will need to contact you network administrator to make sure the firewall is not blocking the 5432 port and/or SYN packets being sent and received through it.

2. Browsing the backend

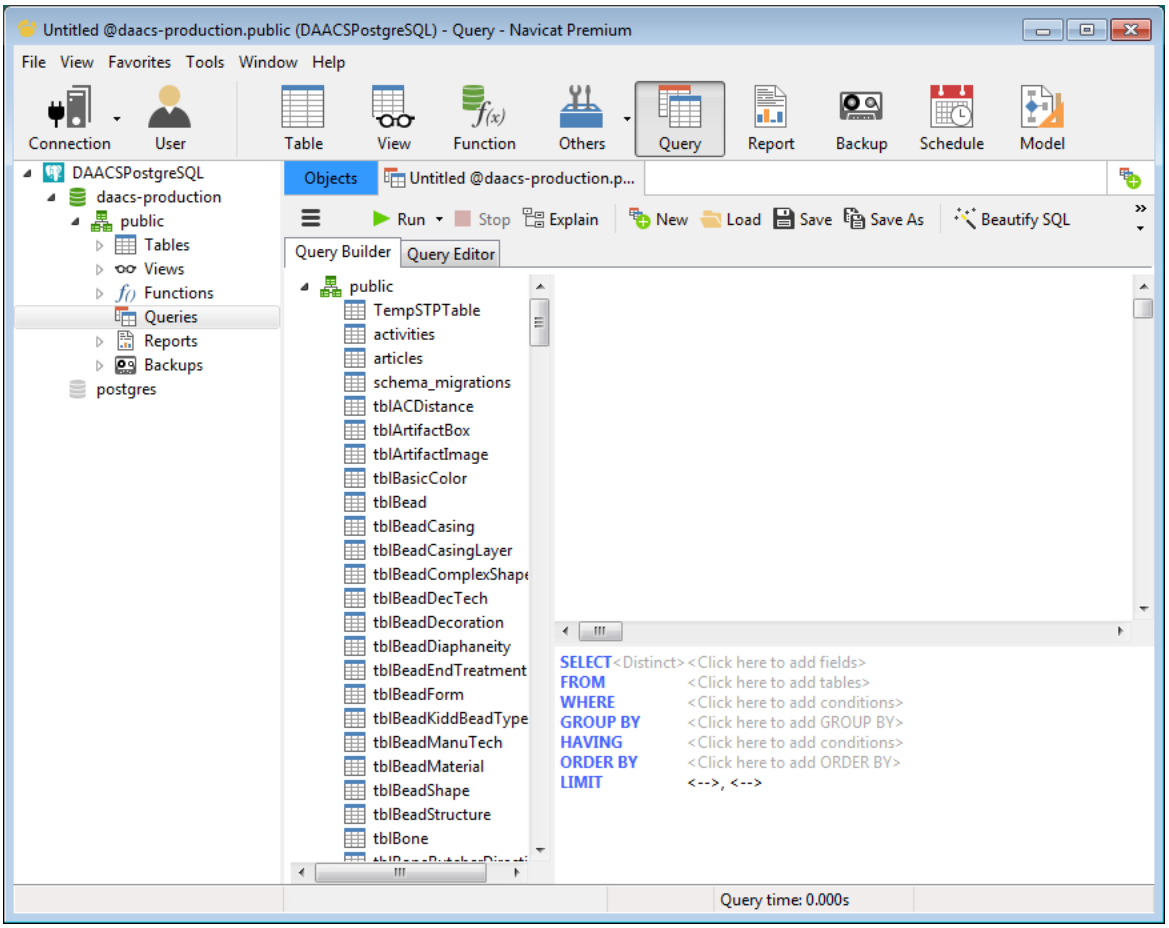
Once you have the connection established, locate the DAACSPostgreSQL icon in the upper left of the screen. *<Double click>* "*daacs-production*" below it, then *<double click>* "*public*". You will see a list of all the data tables in the backend (*e.g. tblBead*). If you double click on the table name, you will see the field names in the table, and the first 1000 records in a new tab. Click the tab to close the table and return to the table list.

Notice the three small icons lined up in the bottom right corner. These control which of three views of the backend Navicat displays. The default is the "*List*" you have just seen. "*Detail*" shows you how many records each table conations. "*ER Diagram*" produces an entity-relationship diagram, showing all tables, the fields they contain. Tables that share the same fields are connected by lines. In SQL these shared fields are used to join tables together, bringing the information from each table together in common result.



3. Query Builder

To access Navicat's *Query Builder*, click the *Query* icon on the top menu bar, then *New Query*. You will see two tabs: *Query Builder* and *Query Editor*. Choose *Query Builder*. You will see this:



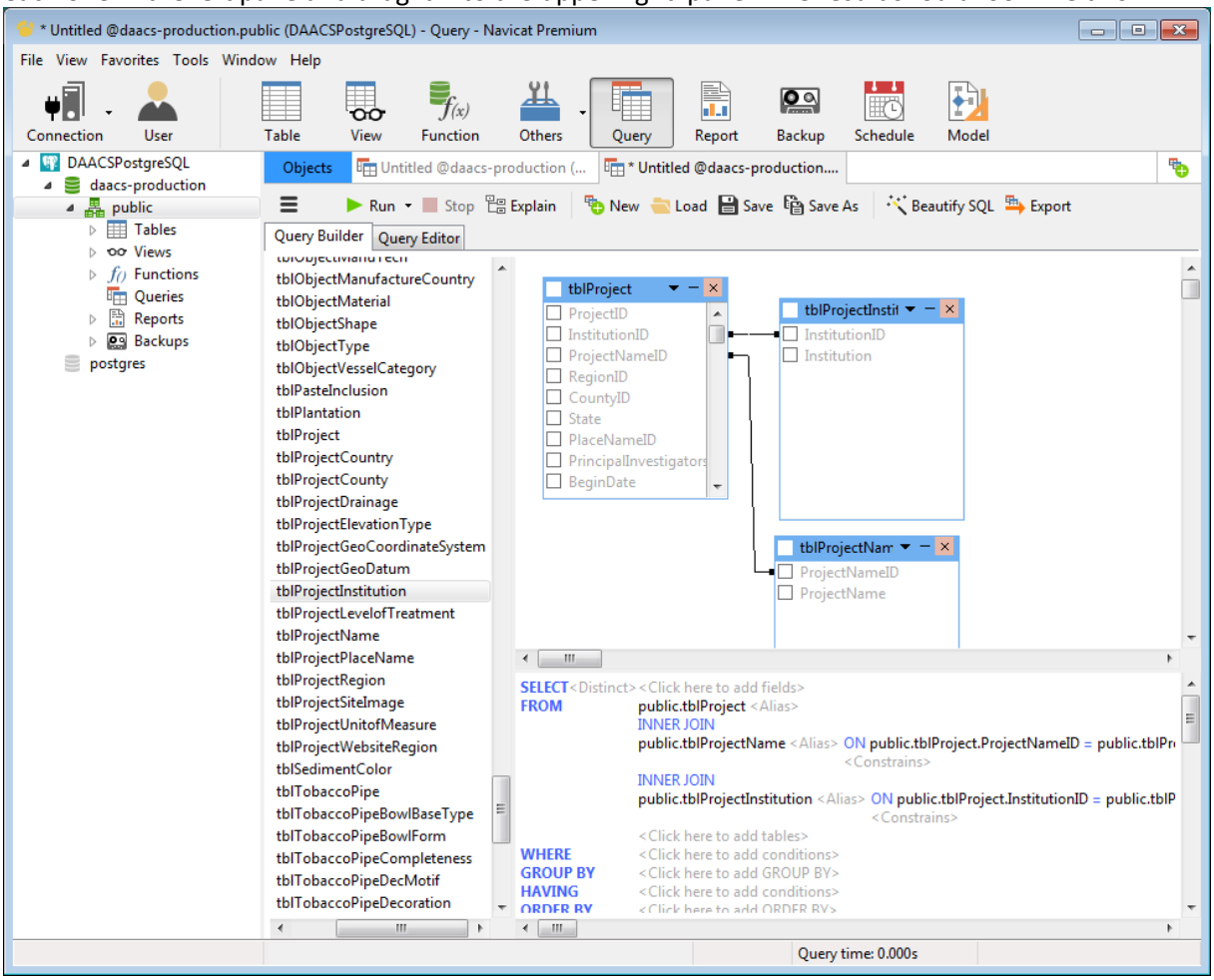
Note the list of the tables in the left pane. These are all the tables in the backend. In the upper-right pane you will find the space in which you will build your query, using drag and drop. The lower-right pane contains the SQL code that your actions in the upper-right pane generate.

4. A First *Query Builder* Example: the SELECT … FROM and JOIN statements

To see how the *Query Builder* works, we need an example. Let's say we want to get a list of all the projects in the DAACS backend. Specifically we want to grab the numeric project ID, the linked project name, the institution. The numeric project ID is useful for running queries that pull out data for specific projects.

We can start building the query by finding the project table (*tblProject*) in the left pane. Select it with the mouse and drag it to the upper-right pane. You will see a list of columns (variables) that the table contains. Any column name that ends in "*ID*" contains numeric codes. The text strings that tell you what the numeric codes mean are stored in separate table -- this is called an "authority table". The authority table contains two columns, one for the numeric code and one for the corresponding text string. For example, the table *tblProject* contains the field *InstitutionID*, which is linked to a separate authority table, *tblProjectInstituitionID*. The authority table also contains the field *InstitutionID* and a corresponding field called *Institution* that contains the text string that is the institution's name.

We'll use two authority tables in our query: *tblProjectInstituitionID* and *tblProjectName*, select each one in the left pane and drag it into the upper right pane. The result should look like this:



Note how Navicat has drawn lines to connect the two ID fields (*projectnameID* and *InstitutionID*) in *tblProjec*t to the corresponding ID fields in the two authority tables. The lines represent the possibility of using these common fields to join *tblProject* to the other two tables, bringing the fields from all three tables together in a new table that has as many rows as *tblProject.*

The next step in constructing our query, is to choose which fields we want in the new table. Do this by checking the boxes next to the field name. For our query, we want to check *projectID* from *tblProject, Institution* from *tblProjectInstitutionID*, and *ProjectName* from *tblProjectNameID.*

Once you have checked the boxes, click the *<Run>* button on the menu bar. You will now see two windows. The upper one is the Query Editor, where Navicat dsiplays the SQL code that you generate with the forgoing points and clicks. Check it out. There are three SQL statements: SELECT, FROM and INNER JOIN.

The SELECT specifies which fields you want returned and the names of the tables in which they reside. The general format is:

```
SELECT  "tableName1"."fieldName1",
        "tableName2"."fieldName2"
```

You can have as many arguments as you want. They must be separated by commas.

The FROM statement specifies the name of the primary table from which the data will be drawn.

The INNER JOIN statement is required here because the SELECT statement asks for fields from more than one table. SQL has to know how to put those tables together (join them). In this case, it needs to know which common fields to use to line up the table rows correctly. It also needs to know how to handle cases in which there is no entry in the ID field of the table named in the FROM statement that matches an entry in the ID field of the table named in the JOIN statement. Using an *INNER* JOIN tells SQL to drop such records.

5. Export Results

The lower window contains the results of the query. You can export the query results in a several formats, but you must first save the query. Once your query is saved, click the *<Export>* button at the right side of the menu bar. The dialog allows you to save the results in .XLS or .CSV files, in the directory of your choice.

6. Saving Query Code

You can save the query code for later use by clicking *<Save As>* and naming it, for example *projectExample1.sql*. Navicat places the code in a folder that it created when you installed it on your machine. The folder is called *Navicat*. It resides in the default document folder on your machine. For me this is the h:\ drive. So the full path to the code file on my machine is: *h:\Navicat\PostgreSQL\servers\DAACSPostgreSQL\daacs-production\public.* The folder structure makes sense:

- *DAACSPostreSQL* is name of the Connection to drc.iath.vriginia.edu server in Navicat
-*daacs-production* is particular database on the server
-*public* is the schema name – a PostgreSQL schema is similar to a folder on your PC.

7. More with *Query Builder*: Sorting

Let's modify the query so that it sorts the results in the *Institution* and *ProjectName* fields. We achieve this by using SQL's ORDER BY clause. <*Click*> the *Query Builder* tab to return to the graphical display of the three tables and their links. Select the *Institution* field within *tblProjectInstitution.* Now <*right click*> and select *Order By* > *Asc* . In *tblProjectName* select the *ProjectName* field. Now <*right click*> and select *Order By* > *Asc* . Click the <*Run*> button on the menu bar. The new results will appear, sorted by the two fields you have chosen.

8. More with *Query Builder*: the *WHERE* clause

Let's say we are only interested in those projects that were undertaken by Colonial Williamsburg. Use the mouse to select the *Institution* field and right click. Select "WHERE" > "IS LIKE". Then in the Edit box type *'%Williamsburg%'*. Note the single quotes which tell SQL this is a text string. Note the % signs which are character wildcards. Enclosing "Williamsburg" in % signs means that any record containing that word in the *Institution* field will satisfy the WHERE clause and be returned by the query.

FDN
6/4/2014

Updated by JEG
7/29/2022