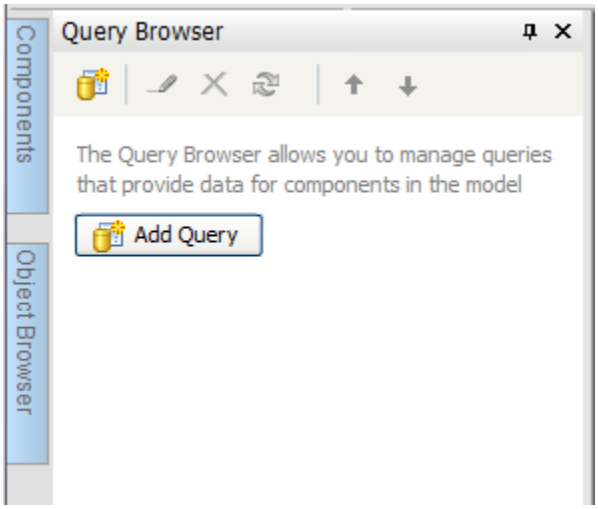
Using Query Browser in Dashboards 4.0: What You Need to Know

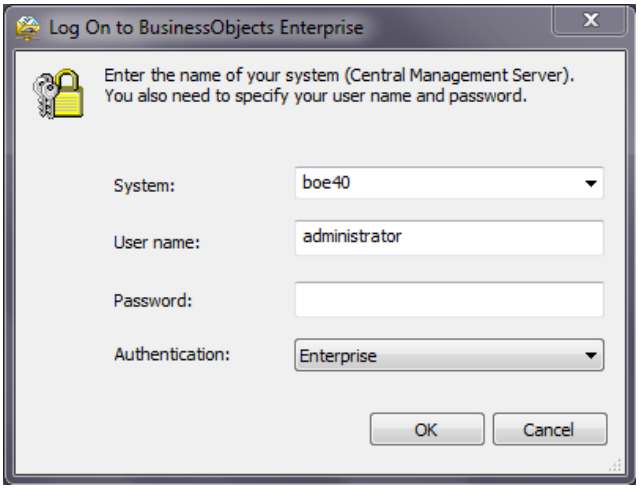
The BusinessObjects 4.0 release gave birth to a few new interesting features to the BI toolkit. One key enhancement was the addition of the Query Browser to the Dashboards (formerly Xcelsius) application. The idea was inherent and simple: allow the developer to obtain data directly from a Universe or BEx Query, thus bypassing the use of the embedded Excel spreadsheet. In this context, the feature achieves what it intends to. However, the feature has its limitations and is not a comprehensive replacement for data connections, such as Web Services. In this post, I will discuss these limitations and what I have found to be the most practical cases for utilizing the Query Browser in your dashboards.

Querying Data Using the Query Browser

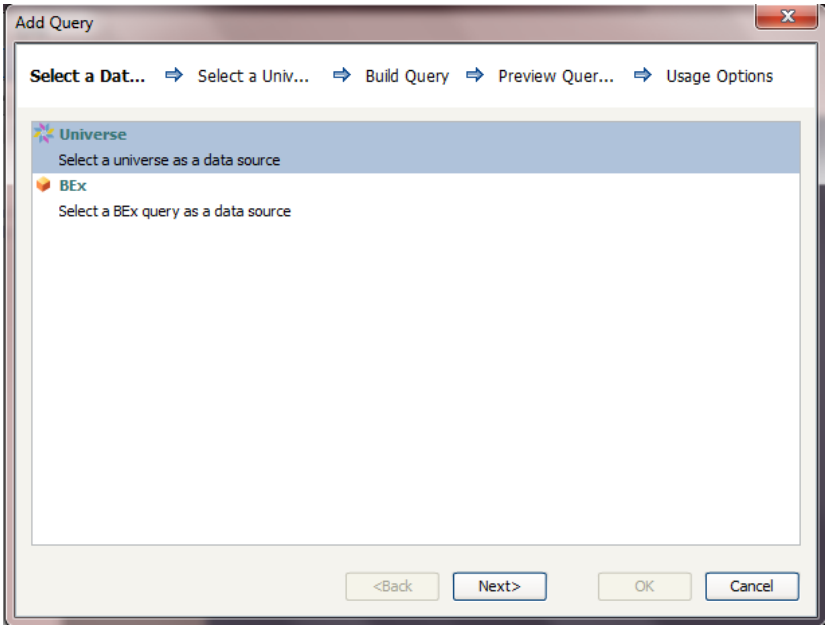
Let's first examine the process of querying data using the Query Browser.



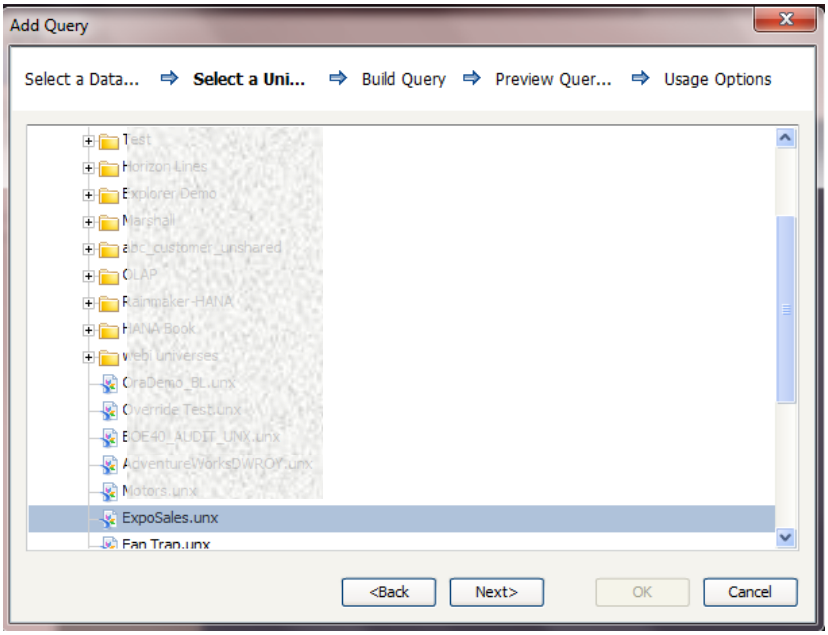
To create a query using the Query Browser, the user must open the Query Browser panel and select either the "Add Query" button (if there is no query already created) or the  button (regardless of prior created queries).



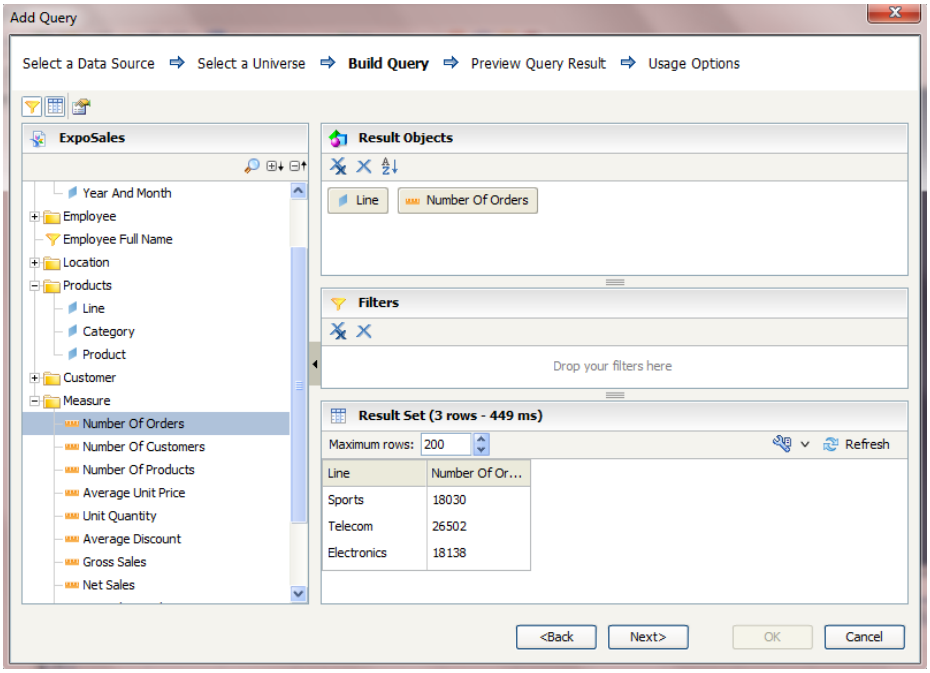
In order to access the correct BusinessObjects system, the user must first input a set of credentials. Note that once the user logs into a BusinessObjects system that is the only system the user will be able to access for the duration of the Dashboards session (the application must be closed to access another system).



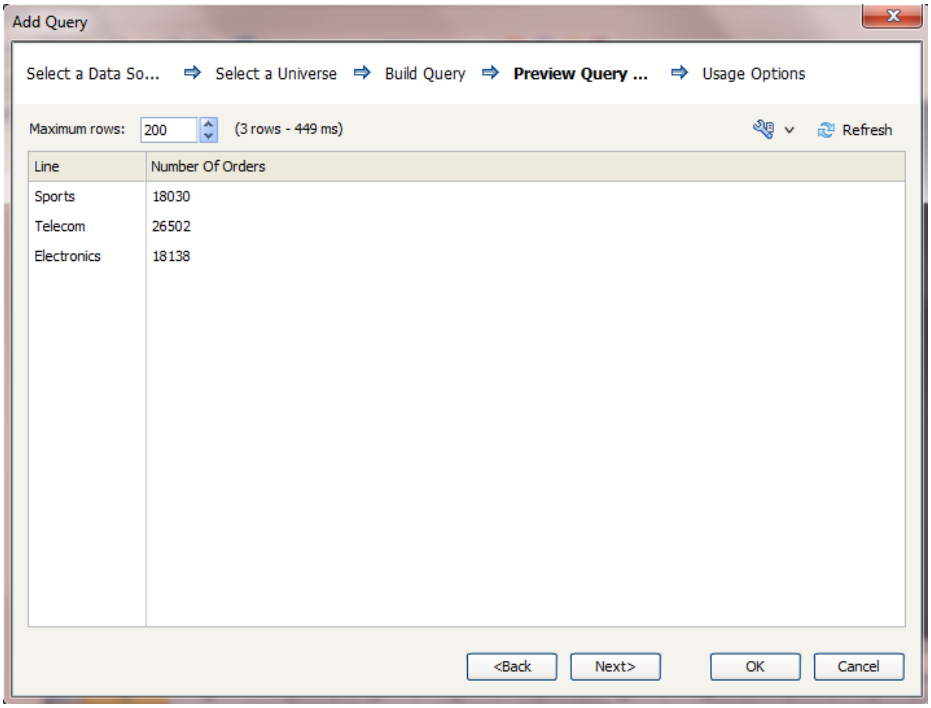
The user starts by choosing to query from either a Universe or BEx Query.



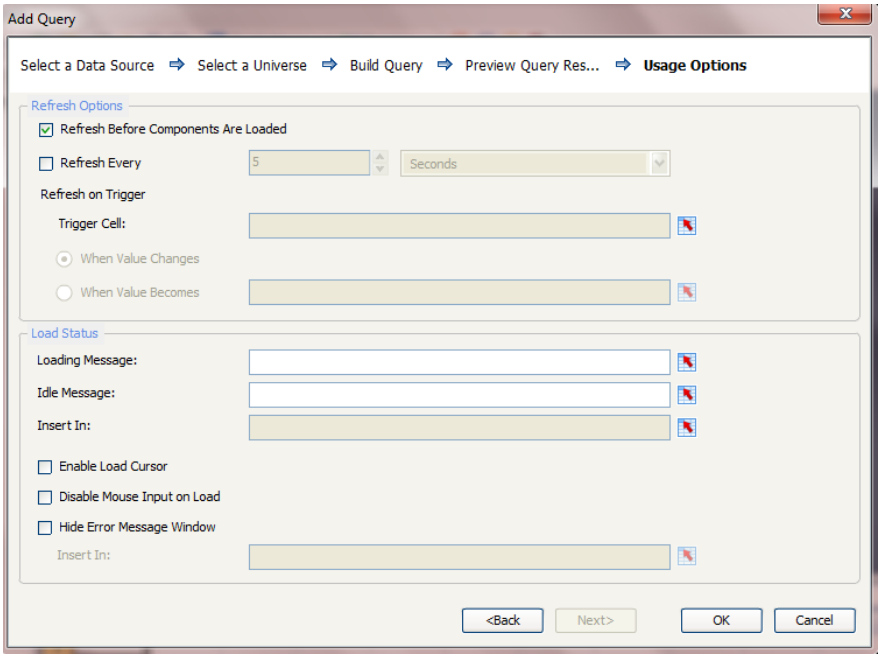
The user is then presented with the list of either Universes or BEx Queries from the BO Repository to choose from.



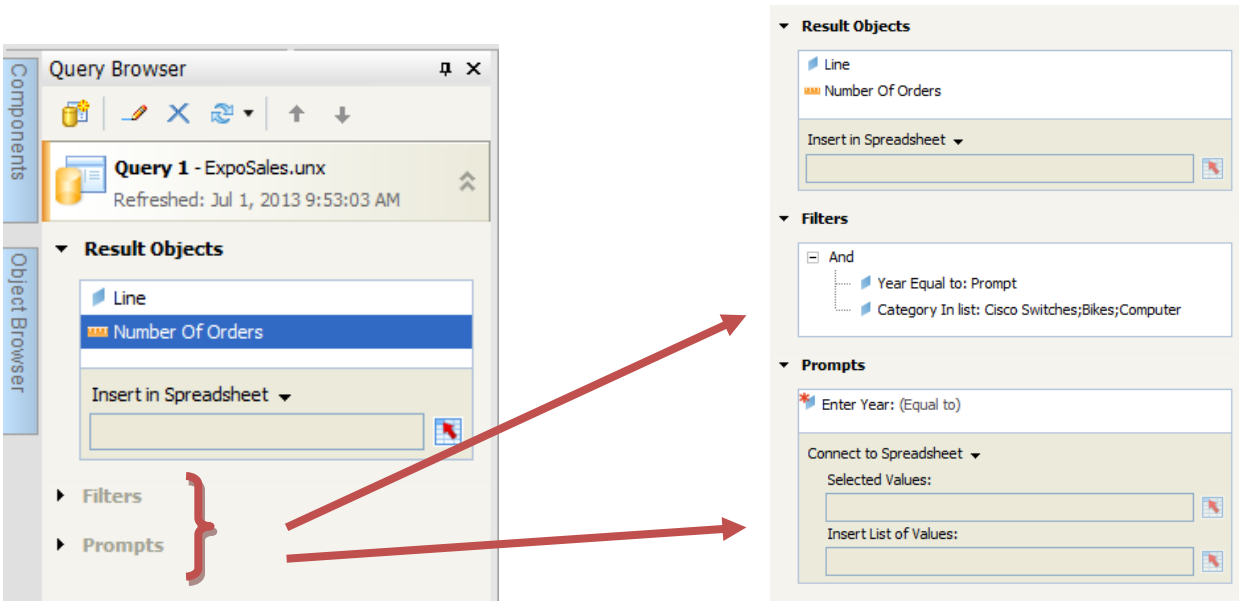
The user is then brought to the Query Panel, essentially a mirror of the Query Panel in Web Intelligence. The user can choose the objects to be returned, apply filters, access query properties, and preview the data.



After the user has made their selection from the Query Panel, the Query Browser presents them with a Preview of the data returned by the query. This display represents the format that the data will take when it is returned by the query.



In this window, the user is presented with several options for Refresh parameters and Load Status displays. When the user clicks OK, they will leave the window and return to the Workspace.



The Query Browser Toolbar now displays the new query, the result objects from that query, and any filters or prompts that the query utilizes.

Using data connections like Query as a Web Service (QaaWS) incorporates the following steps:

1. Creating a dataset
2. Setting up a data source connection
3. Setting up a connection to the data source in the dashboard
4. Defining target cells and dropping the data into the spreadsheet
5. Binding components to the data

As you can see, this is a pretty cumbersome process with the potential for issues arise. Compared to the setup process of the Query Browser, great care must be taken to ensure that all steps in the process have been defined correctly. The beauty of the Query Browser is that these steps are simplified and consolidated, making development faster and easier. As an added bonus, the Query Browser allows the user to do all but the final step from a single wizard.

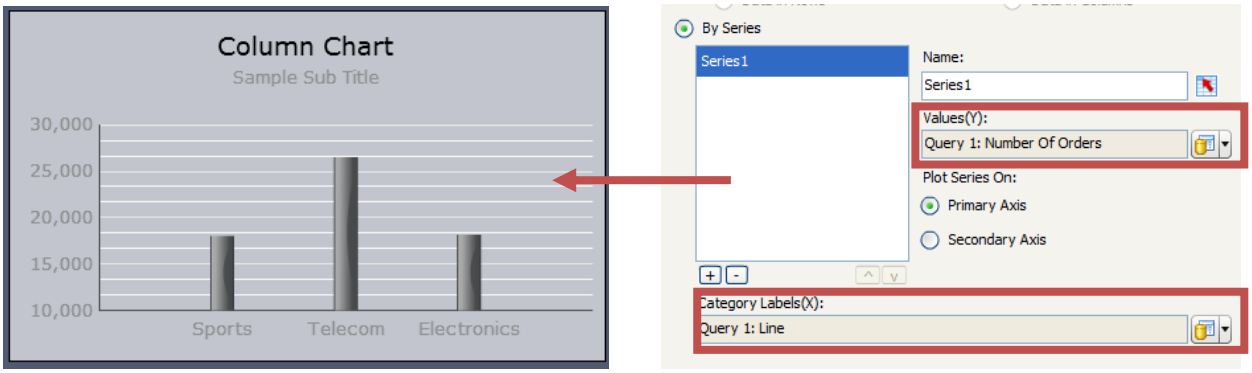
Using Query Browser Data

In order to utilize the data obtained by the Query Browser, the user has two options. The user can simply drop columns of data from the query into the Excel spreadsheet or bind data directly to components. The former is useful when transformations to the data are needed, the latter for simplifying the component setup and reducing the load on the spreadsheet engine. Practiced developers are certainly aware that the inclusion of large datasets in the spreadsheet can severely impact dashboard performance at runtime.

When dropping data into the spreadsheet, the user needs to be conscious of the format of the data. The user needs to recall the format displayed in the Preview stage of creating a query to be sure of the

tabular form that the data will take on as it enters the Dashboard application. Also, dropping data into Excel must be done one column at a time (note that these columns of data will contain non-unique values). This adds extra steps and can be time consuming for queries of several columns. However, if the data does not need to be in one continuous range of cells, these columns can be placed wherever they may need to be located. This can be very helpful under the right circumstances.

Like dropping data into the spreadsheet, binding data directly to components requires attention to the format of the data being bound. The data is stored in a tabular view and thus loses the dynamic exploration of a microcube, like Web Intelligence. Therefore, components will display every row/column of the data explicitly as it comes into the application. Datasets that are bound directly to components should be simple enough (typically only one dimension) to reduce the complexity within the aggregation of measures. Unnecessary dimensions should not be included in the result objects of the query.

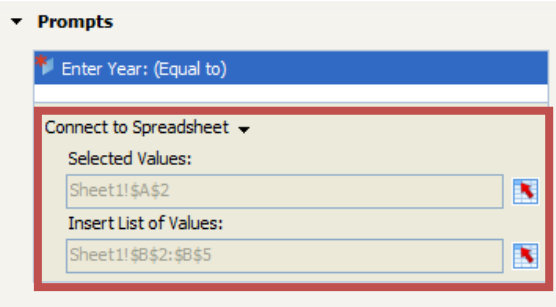


In both of the above circumstances, a somewhat overlooked feature (that Web Services allows) will now prove to be more difficult with the Query Browser: one-time manipulation of data. The query produced from the Query Browser is entirely dependent on the objects available in the Universe or BEx Query. Therefore, any data manipulation must take place in the database (or cube), in the Universe or BEx Query, or in the spreadsheet within Dashboards. Since the greatest advantage of using the tool is that you can bypass the spreadsheet, dropping the data here is not recommended—unless the manipulation taking place is very simple and on a small dataset. Requiring the spreadsheet to perform the manipulation will also drastically affect performance.

The issue with pushing this manipulation down to the database or Universe or BEx Query is that this likely will require involving an extra party in the dashboard development process (unless the developer has access to make these changes, that is). In situations that call for data manipulation as a method for creating a one-time fit, QaaWS from BusinessObjects is the better solution. This is preferred because QaaWS references the data in terms of columns instead of objects, and the process of creating formulas in Web Intelligence is rather simple. The data can be easily manipulated there, and the way it is projected into the report block is exactly what will be obtained through the Web Service.

The Query Browser has versatility when incorporating query refreshes. The tool allows the user to bind prompt value inputs and prompt list of values outputs to cells in the spreadsheet. This is a valuable functionality as the dashboard design may necessitate changing the prompt values to obtain subsets of

the data. This eliminates that need to pull in more data than is needed. This is another area where Dashboards reduces the drag the query would otherwise put on refresh time.



Conclusion

The Query Browser is a valuable tool under the right circumstances. In my opinion, binding data to simple charts, dropping small sets of data to the spreadsheet, and obtaining lists of values for selectors are the most relevant use cases. The tool—in its current incarnation—will not be a complete replacement for Web Services, due to its limitations. But, the Query Browser was a thoughtful addition and will certainly relieve a few pain points that were present before its inclusion. I will certainly get a great deal of use out of it.

James T. Mason

Business Intelligence Consultant

Decision First Technologies

James.Mason@decisionfirst.com

James Mason is a business intelligence consultant specializing in report, dashboard, and semantic layer development. James delivers customized SAP BusinessObjects solutions for customers across all industries. With Decision First Technologies, James utilizes Web Intelligence, Dashboards, Universe Designer, Information Design Tool, and Explorer.