DDBuilder™ - New GUI Data Dictionary Editor

By Dennis Tokoph

Business Basic applications revolve around databases. Large or small, newly created or legacy system - they all have a common theme: the information is stored in a database containing multiple tables, columns, and indices. This also means that they share a common problem when the time comes to retrieve that information using SQL and ODBC: how to tell the BASIS ODBC Driver™, PRO/5™, or Visual PRO/5™ what information is contained in each table, column, and index. The answer is to create a data dictionary that describes the database. DDBuilder makes this task quicker and easier by providing a graphical representation of the dictionary, a familiar Windows interface to create it in, and the new views introduced in Version 1.1 of the BASIS ODBC Driver. If you've used the Windows Explorer (File Manager) you'll have no problem working with DDBuilder.

DDBuilder is a Windows utility for creating and maintaining data dictionaries. It comes packaged with the new Visual PRO/5 and ODBC Driver products. You still have the option of using the `_ddedit.utl` utility. (In fact, that is the only choice if you need to develop on a PRO/5 system.) We think you'll find DDBuilder is a real productivity booster. The ability to open multiple dictionaries makes it a simple matter to copy tables, columns, typedefs, rules, and views from one dictionary to another. Or, if you have columns that need to exist in multiple tables, just define the columns in one table, create the other table objects, and copy the columns from the first table to the second. If you're working with non-normalized data, then you're really going to appreciate the new views feature. You can't create views with `_ddedit.utl`.

So what are these *views*? One way to think of views is as individual databases of normalized data created/extracted from a non-normalized database (but without the hassle of actually extracting the records into individual temporary files). They are pointers to the tables, columns, and record types associated with Where statements to select a single record type for each view. So there's no disk space consumed by temporary files. A view can be used in the definition of another view, so you can use them like building blocks to share the most commonly used record types.

The DDBuilder icon in your BASIS folder is the place to begin. You'll see a window similar to Figure 1. Actions on the menus are accessible in a variety of commonly used ways.

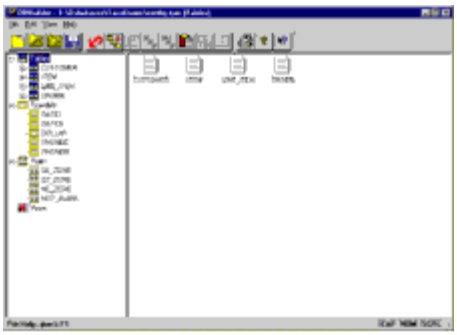

Figure 1. Sample DDBuilder Screen

The left side of the window is where the graphical tree representation of your data dictionary will be displayed when you open or create one. The right side of the window will display property sheets when you open an item in your data dictionary. The divider bar between the left and right side is adjustable if you need to see more of one side than the initial placement allows. I find that moving the bar farther to the left, so it covers part of my object names, is helpful on a 640x480 VGA laptop. When I need to see an object's full name I just place the mouse pointer over it and pause for a second or two. The full name pops up in a tool tip and I'm never left guessing what I'm pointing at.

To create a new data dictionary, select New from the File Menu. This will bring up a dialog box with fields to name your `config.tpm` file, the paths to the dictionary and data directories, and the optional Data Source name that you want added to the `sql.ini` file. Full pathnames are best here, though you can use global variables if that's desirable for your particular application. You should create the directories you want to use before you click on Create. You'll be asked to confirm that you want to create the dictionary files in your dictionary directory. When you click on the Create button the dictionary files will be created and the left side of the split window will be populated with a number of objects - Tables, Typedefs, Rules, and Views. These objects will be empty until you add something to each type. Note that your application may not need to use them all.

To open an existing dictionary that's not yet listed in your `sql.ini` file, select Open Config from the File Menu. Use the Open dialog to navigate to your configuration file and select it. The left side of the window will be populated with the same four objects as above, but now the objects will contain other objects representing your existing Tables, Columns, Indices, Typedefs, Rules, and Views.

To open an existing dictionary that's already been added to your `sql.ini` file, select Open Data Source from the File Menu. A window containing all of the Data Sources listed in your `sql.ini` file will appear. Double-click on the Data Source you want to open. The left side of the split window will be populated immediately with the objects listed above and they will be filled with your existing Tables, Columns, Indices, Typedefs, Rules, and Views.

To add a new Table to the opened dictionary, select the Tables object in the tree. Then select Add from the Edit Menu. A basic table object will be added to the tree under the Tables object with a generic name. To rename the table, click on the name and replace it with the desired name.

The property page is presented in the right side of the window.There are multiple property pages for entering the file type, default form, and other details related to the table. Select a tab at the top of the form to change pages. You'll want to pay particular attention to the file name, file type, and default form name to insure everything is named as you desire.

Figure 2. Sample Property Page

Next you should add columns to the new table. Make sure that the new table is highlighted by clicking on its icon, then click on the Add Button. A basic column object will be added to the tree under the table object you created above. Again, you can change its name to whatever you desire. Once the name is set, bring up its property pages. There are multiple pages upon which you can set the field type, size, label, etc. Special types can be selected if you have already added them to the dictionary. (You can define Typedefs and Rules before you start defining tables and columns to increase your efficiency.)

Once you have defined columns for your table, you should define indices. Select the Indices object and click on Add. This will create an index base object. You'll then need to select the base object for two further actions - to rename it and to add Segments to the index. For SQL and ODBC access you'll want to make these *dependent* segments. Dependent segments consist of one or more columns named in the table. (Though independent indices are supported when running pure Business Basic applications, the SQL and ODBC languages don't understand them. Use only dependent indices if you intend to use them with SQL and/or ODBC queries.) To add more than one segment to the index, highlight the index base object and select Add again.

Everything presented to this point can be done with the `_ddedit.utl` utility. TAOS allows you to do more, such as adding special typedefs and data validation rules to the data dictionary definitions. These features help you standardize data between applications and insure correct data gets entered before it is saved to the database. DDBuilder carries on this tradition of increased efficiency.

Adding Typedefs and Rules is the same process as adding Tables, Columns, and Indices. Select the Typedefs or Rules object and click on Add. Change the name as desired and then select the new object to access its property pages. Each type of object has appropriate pages for setting the details of the Typedef or Rule. Edit the fields on each page as necessary.

Views are added in the same manner. The property pages for a view are simple to define. Enter the Table and Column names that the view will select data from and a Where statement that selects the proper record type from the Table. You can even add other criteria for the selection such as limiting the selection to orders under 1000.

Once you have everything defined, you should save your dictionary. If you are working with a new dictionary, you can create the empty data files with all of your information. **A word of caution - don't use the Make option on pre-existing data files unless you want to remove all records from the files. Make creates empty data files only.** You will be reminded of this when you click on Make.

Your productivity will increase when you have a new dictionary to create based on pieces of existing dictionaries. By running multiple instances of the DDBuilder you can copy an item from an existing dictionary to a new one. This can save you a significant amount of time and effort. It will also insure that typos don't get introduced, so the same definitions will get used consistently across your dictionaries.

Copying an object from one dictionary to another is a simple task. After you have one dictionary opened you don't have to return to the icon in the BASIS folder to start a second session. Opening another data source will automatically create a new session. Once both dictionaries are open you just select (highlight) the object you want to copy then choose Copy from the Edit Menu. In the other dictionary window select the proper place for the object to be copied to and choose Paste from the Edit Menu. The object will be added to the new dictionary.

We think you'll find DDBuilder is a perfect companion for Visual PRO/5 and the BASIS ODBC Driver. It is one more way that BASIS keeps giving you the best business application development tools in the industry.