New List Box Features in 4D v11 SQL

By Larry Sharpe, 4D Developer, InfoService

Technical Note 07-38

Introduction

4D v11 SQL has added several new features for working with List Box form objects. A List Box is quite useful in many ways but was limited to only using arrays in previous versions of 4D. Now you can not only use arrays, you can also use fields and expressions to populate the List Box. This Technical Note presents three examples that are very similar in how they appear to the user, but each uses a different "Data Source" for populating, displaying and using the data in a List Box.

Data Sources

The possible Data Sources that can be used with List Boxes are:

- **Arrays** – Same as in 4D 2004.
- **Current Selection** - Can now show fields and expressions based on the current selection of records you are working with.
- **Named Selection** - Similar to the Current Selection, but based on using a Named Selection (an ordered set of records).

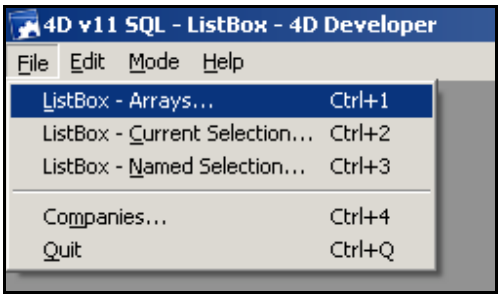
The example database contains examples for all three Data Sources.

Examples

The example database has two tables, [People] and [Companies], with a few standard fields for each table. The ID field in each is automatically set using the Trigger for each table. Each table has a basic Output and Input form that can be used in “User Mode” (now called “List of Tables” in 4D v11 SQL) as needed. The two tables are related in that [People] records have a link to the [Company] record that they are associated with. This simple design represents a straightforward, standard set of features of a simple database.

One new feature of 4D v11 SQL is that you can build and use forms that are not tied to a specific table. These new forms are called **Project Forms** and are used in the four example forms that will be covered here. This is a great place to store forms that are not tied to a specific table (e.g. dialogs).

The example database offers the following 4 demonstrations (under the file menu):



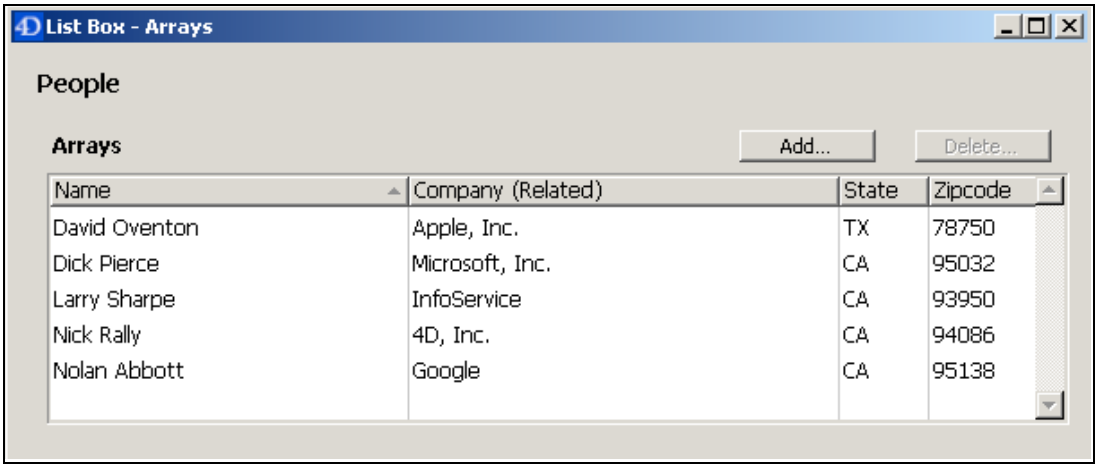
Each of the three types of List Boxes use Project Forms that are all very similar in look and functionality. You have a List Box which shows columns filled with data from the [People] and [Company] tables. You can Add, Modify or Delete [People] records by double-clicking a row (modify) or selecting a row and clicking the Add, or Delete buttons. You can change the code if needed to allow for selecting multiple rows of the List Box, but this demo has it set to only allow a single row to be selected at one time. Setting to multiple row selection is as simple as changing the List Box Properties and the Delete button script.

In all of the examples we are using the form editor to select which tables, fields, and other settings we want to display in the List Box. For 95% of the databases that you develop this is fine, but you could do this by using List Box and Object Properties commands to set this all up in code. There are several Technical Notes available that will show how to do this.

Besides the Project Form for each type of List Box we also use a Project Method named ListBox_xxx_Functions for the code that is used several times by the List Box's Object Method and the Add and Delete button’s Object Methods. Instead of having multiple copies of the same code all over the place, we put it this in one method and call it when needed.

Note that the source code in the example database is fully commented to provide more information about what is happening when it is executed.

Example – Arrays List Box

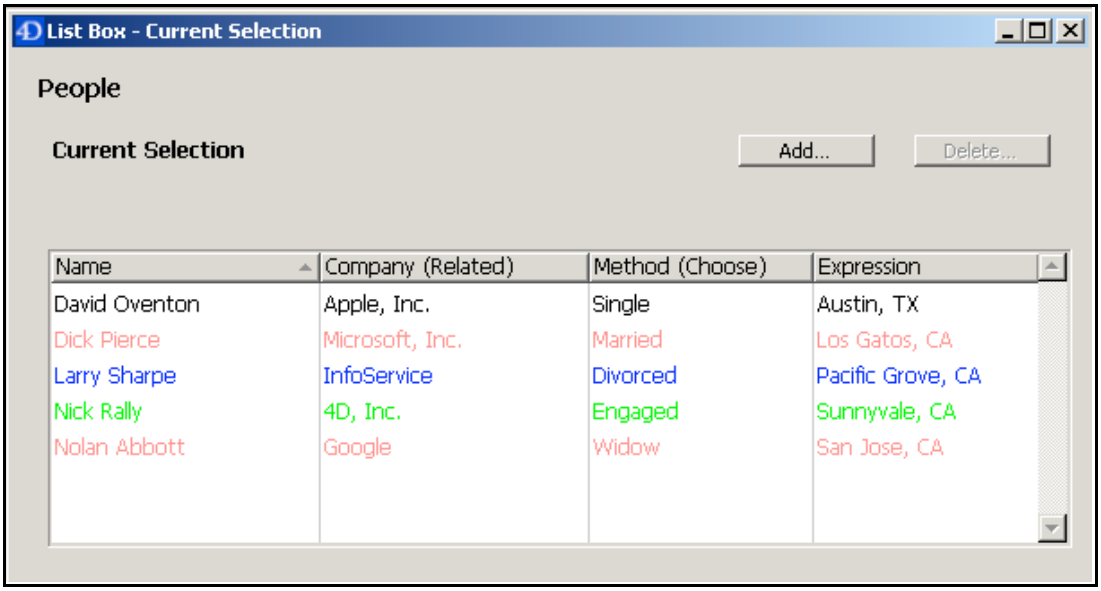


Here we use arrays that will be shown in the List Box. This is the same as how it works in 4D 2004; you use a **Selection To Array** command that loads the appropriate records into arrays and you define which arrays are used for which columns. Once the List Box is displayed, you can double-click a row and it will do a Query to find the correct record and change the display to allow the user to modify it. Deleting a row is very similarly, find what the user clicked and ask if they really want to delete it. Everything is based on loading the arrays to be used and then querying to find the selected record to modify or delete it. Note that you can use a related table's data to populate an array if needed.

We do load one array that is included in the List Box settings but is marked as invisible to the user. This is the [People]ID field which is used to identify and find each record when needed. The other Data Sources do not need this column since they are using the fields directly from the table and thus know which records are being clicked or double-clicked.

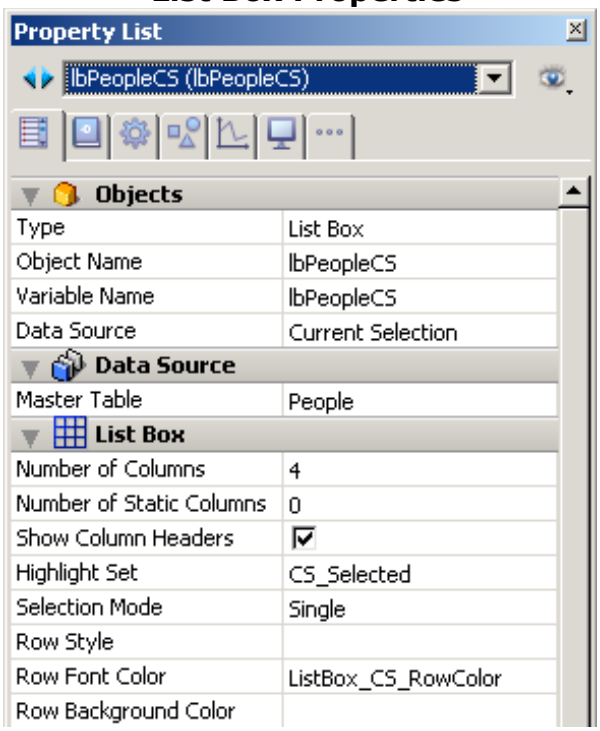
Another thing that the "Arrays" Data Source does not have is the ability to use Expressions directly in the List Box. You could do something similar when you load the arrays with data, but it can not be entered in the column itself like the next two Data Sources will show.

Example – Current Selection List Box

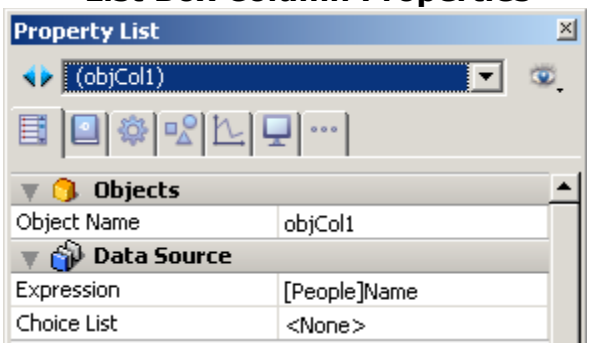


In this example we are using the current selection of [People] records to be displayed in the List Box. It looks very similar to using arrays except we do not have to define, load and clear arrays. This is the key difference in 4D v11 SQL; to setup the List Box, you select which fields for which columns (when building the form) in the Property List:

List Box Properties



List Box Column Properties



In this simple example, we are just doing an **All Records** command, but you could do a Query to find a specific selection of records or a Relate Many to load all the related records of another table.

Besides selecting which field to display in which column, you can also use an Expression to fill a column. The expression can be as simple as the following code:

```
[People]City+", "+[People]State
```

This can be entered directly in the column's Expression Property or you could enter a Project Method name which could do a much more complicated bit of code to return what each row of data should use for that column. The Expression will run separately for each record being shown in the List Box when it is shown, you do not need to loop through the records in code to handle this, it is done for you.

To show how to use an Expression using a Project Method we use another new 4D v11 SQL command, **Choose**, which allows you to simplify the code when displaying data that has been selected from multiple choices. For this example we have a Longint field in the [People] table named [People]MaritalStatus. Instead of showing the number, which is not that useful to a user, we want to show the appropriate text value that the number is associated to. Using the Choose command we pass the field name and the appropriate text values that can will selected based on the number passed. Think of it as a simple (one line of code) way of doing a **Case Of** statement which has multiple lines of code to support all the various values. Here is the example code (this should all be on one line, in 4D, of course):

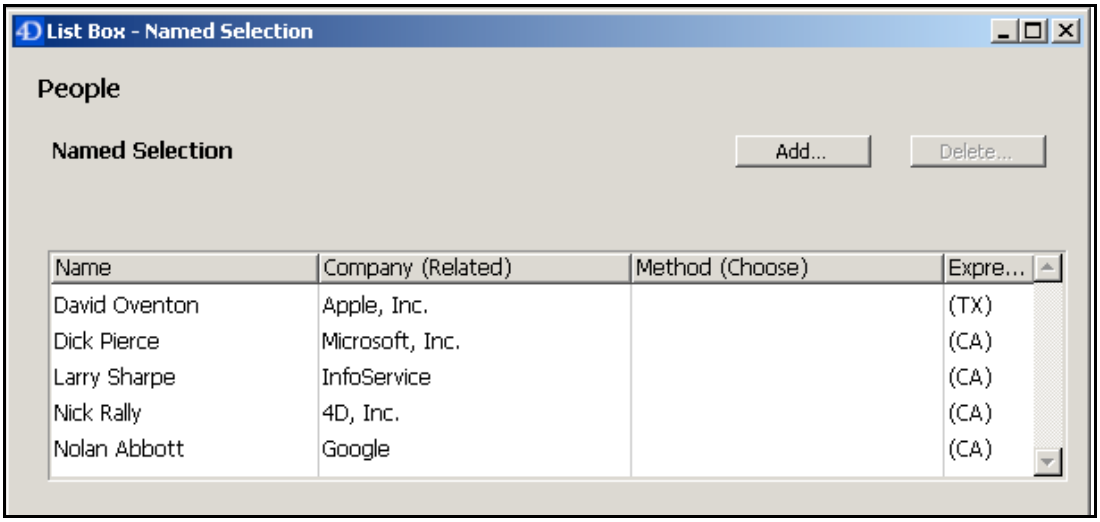
```
$0:=Choose([People]MaritalStatus;"";"Single";"Engaged";"Married";"Separated";"D
ivorced";"Widow")
```

In the Input form we use a pop-up built by doing a **List To Array** command to show and select the value stored in this field.

Another nice feature available when using Current Selection (or Named Selection) is that you could allow the user to modify the data directly in the List Box cells. This can be done by typing in a new value or picking from a pop-up array. When using this feature, make sure that the table is set to READ WRITE mode and the appropriate check box is set in the List Box settings.

Unlike the other two Data Sources, when in the Input form you can use the Previous/Next buttons on the Input form.

Example – Named Selection List Box



Using the Named Selection Data Source is very similar to the Current Selection. Named Selections are similar to Sets except it keeps the sort order and current record of the selection maintained for you. Other than for a few things described below, all the programming and interface features of a List Box using Current Selection are available to you.

You could create a bit a code that allows you to sort the Named Selection being shown in the List Box that is very similar to that shown in the Current Selection example, with the exception that the user interface does not show which column is used for the sort. Sorting this type of List Box seems a little excessive since part of the reason for using Named Selection is that the sort order is preserved, but you could do this if needed. It is not done in this Technical Note example.

In this example database if you delete a record from the Named Selection List Box it will reload the records and recreate the Named Selection

Other Example Database Features

Sorting Columns

In the Arrays and Current Selection examples we are using the **SORT LISTBOX COLUMNS** command instead of the **ORDER BY** command. We are also turning OFF the Sortable Property in the List Box Properties. Be careful not to use the SORT LISTBOX COLUMNS command if the Sortable Property turned ON as this may produce unexpected results.

You can not sort on a column that is using Expressions.

When using Related fields in the Current Selection mode the column sort is done on the Master Table's field value not the Related table's field value. In the example database it sorts on the [People]CompanyID field and not the [Company]Name field that is shown in the List Box.

When using arrays, keep in mind that it is just sorting the arrays that were loaded at the beginning and not the actual records.

Highlight Set

When setting up either a Current Selection or Named Selection List Box object on a form, you need to enter a name of a Set to be used when a record is highlighted by the user. You do not have to populate the set, as that will be done by the List Box, but you do have to name it in the Property List for that List Box. If this is not done then the user will not be able to highlight any records in the List Box and you can not do it with code. When using Arrays a Boolean array is created automatically just like in earlier version of the List Box.

Style and Fonts Settings

When using Arrays you use the same method as before, you fill the appropriate arrays with your settings for Styles, Font Colors and Background Colors, i.e., each element of these arrays matches each element of the data arrays. When using Current Selection or Named Selection you set these values in a manner that is very similar to using Expressions. You use a Project Method that is called for each row individually and it sets the specific Styles or Fonts depending on what is in that row (record). If you would like to set all Styles and Fonts to be the same it is not necessary run code for each row, you can set the default values in the Property List settings. Take a look at the Project Method *ListBox_CS_RowColor* to see how we set the row color based on each rows (records) [People]MaritalStatus value.

Conclusion

The List Box functions in 4D v11 SQL now have three ways of populating the List Box. There is the extremely useful way of using Arrays that was part of 4D 2004 and two new ways by using either the Current Selection or Named Selection of records. This mean that you do not have to load, manipulate and clear arrays. You can use records and expressions directly within the List Box. This allows you to easily set up and use List Boxes with little, if any, code.

Larry Sharpe can be contacted at (831) 373-6266 or LSharpe@infoservice.com