Preprint
UCRL-JC-136117

# WebDB Component Builder – Lessons Learned

C.M. Macedo

This article was submitted to
*International Oracle User's Group America's, Anaheim, CA, May 7-11, 2000*

U.S. Department of Energy
Lawrence Livermore National Laboratory

**February 15, 2000**

# WebDB Component Builder – Lessons Learned

Charalynn M. Macedo

Lawrence Livermore National Laboratory
UCRL-JC-136117

Presented at IOIUG-A 2000
Anaheim, CA
May 7-11, 2000

# WebDB Component Builder – Lessons Learned

Charalynn Macedo
*Lawrence Livermore National Laboratory*

## Introduction

Oracle WebDB is the easiest way to produce web enabled lightweight and enterprise-centric applications. This concept from Oracle has tantalized our taste for simplistic web development by using a purely web based tool that lives nowhere else but in the database! The use of on-line wizards, templates, and query builders, which produces PL/SQL behind the curtains, can be used straight "out of the box" by both novice and seasoned developers.

The topic of this presentation will introduce lessons learned by developing and deploying applications built using the WebDB Component Builder in conjunction with custom PL/SQL code to empower a hybrid application. There are two kinds of WebDB components: those that display data to end users via reporting, and those that let end users update data in the database via entry forms. The presentation will also discuss various methods within the Component Builder to enhance the applications pushed to the desktop.

The demonstrated example is an application entitled HOME (Helping Other's More Effectively) that was built to manage a yearly United Way Campaign effort. Our task was to build an end to end application which could manage approximately 900 non-profit agencies, an average of 4,100 individual contributions, and $1.2 million dollars.

Using WebDB, the shell of the application was put together in a matter of a few weeks. (Manager's love this!) However, we did encounter some hurdles that WebDB, in it's stage of infancy (v2.0), could not solve for us directly. Together with custom PL/SQL, WebDB's Component Builder became a powerful tool that enabled us to produce a very flexible hybrid application.

## Lesson One – Use of Stored Source:

WebDB on Stored Source

Oracle WebDB provides multiple methods for creating forms and reports, however, when more detail or robust manipulation of data is required, the WebDB wizardry or basic SQL query methods will not suffice. The developer must return to the trenches of coding by hand; however, not all is lost! Oracle WebDB provides a smooth solution, by providing a means to build the interface between your stored source and your end user. This can be extremely useful for accepting input parameters from a user to either manipulate data on the back end, simply display data back to the user, or pass these parameters to another WebDB component for further processing.

A good example of a component built on stored source was the need for a robust report which involved several nested cursors and some basic mathematics. The stored procedure accepted two parameters from the user and displayed the results back to the user via PL/SQL-HTML. The interface was built using the Form Builder.

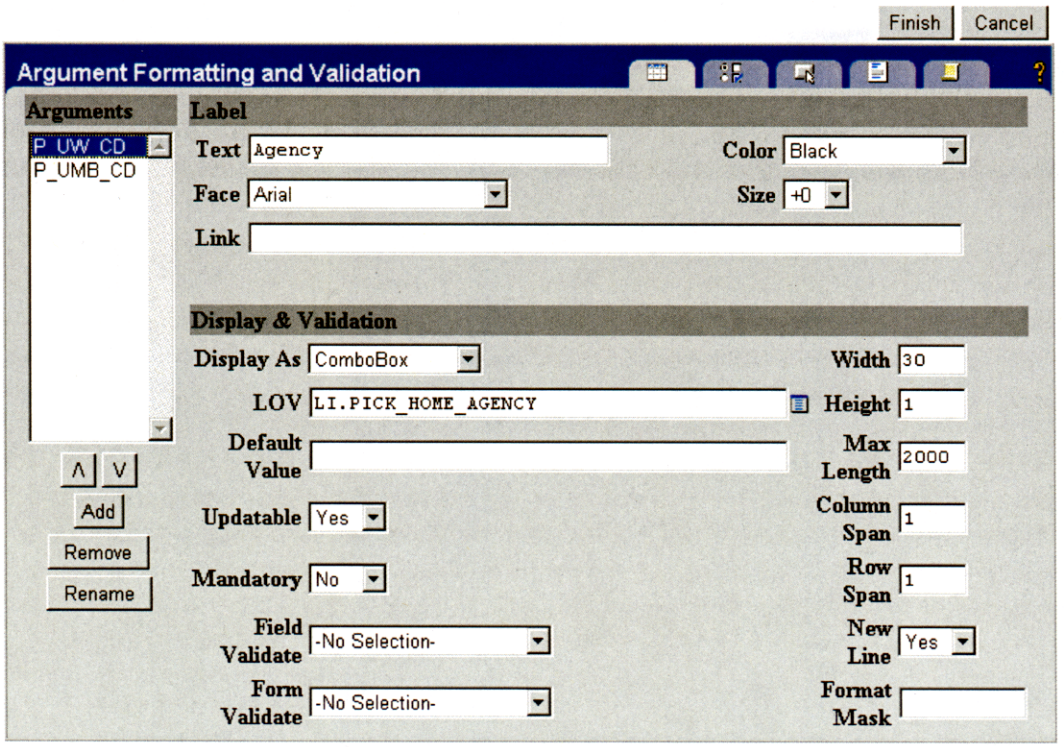



Figure 1. Form Building on a stored procedure. The two arguments shown here, are the two variables named as the input parameters within the stored procedure.

```
Create or Replace Procedure LI_AGENCY_DTL_BY_EMP_PR (
        P_uw_cd   in number default null,
        P_umb_cd in number default null)
```

Complete the Form Builder wizard for GUI requirements and the final result is the input form, which upon submission calls the stored source. The look and feel of the output generated by the custom PL/SQL code is exactly the same as if WebDB had generated the final output. This is possible by cutting and pasting the HTML from the WebDB template directly into your custom code. The result of the hybrid is absolutely transparent to the user.

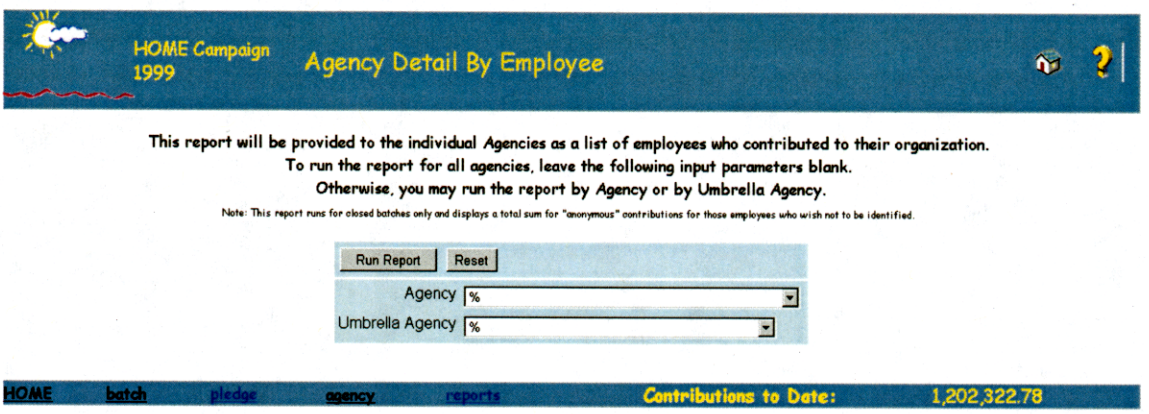


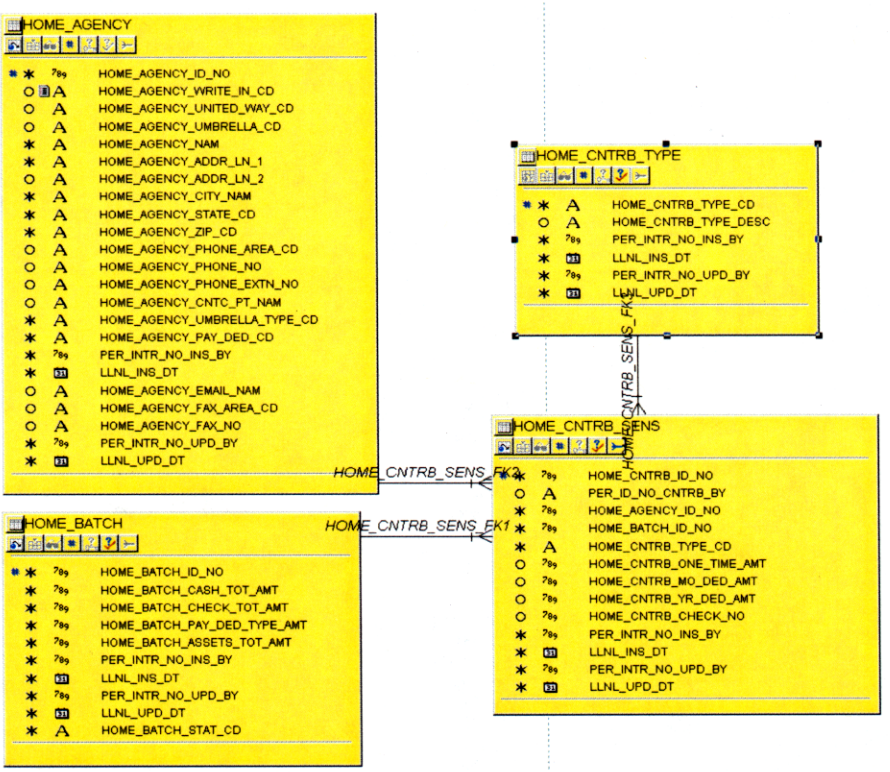

Figure 2. WebDB Form built on a stored procedure.

Note: One drawback to having multiple WebDB components built on stored source is compiling! The stored source must obviously be compiled and in a valid state for the WebDB component to compile. Modifications to these components can be tricky. Any change to one side or the other invalidates both objects in the database.

## Lesson Two – Master Detail:

WebDB Forms and Reports

Another good example of a component built on stored source was a more complicated need to solve a master detail relationship. It could not be accomplished using the WebDB canned Master Detail Form Builder due to the fact we needed to join multiple tables. Below is the relationship we needed to satisfy.




In this example, the user is required to enter an employee id and batch number in order to enter one or more contributions. A small procedure was written which accepts two parameters and then calls the next WebDB component passing the values in a PL/SQL table. WebDB passes values into a PL/SQL table by name/value pairs. The procedure below accepts two parameters through a WebDB form interface (Figure 3). It then calls another WebDB form passing the same values that are required in order to create the detail element. The form shown in Figure 4 doesn't actually display these values but accepts them as part of the insert statement into the base table the form is built on.

```
procedure home_call_contrib_form_pr (
    p_batch         in varchar2,
    p_empno         in varchar2 default null
    )
is
begin
    li.li_home_contrb_form_pg1.show (
         p_request => null,
         p_arg_names => webdb.wwv_standard_util.string_to_table2 (
             'HOME_BATCH_ID_NO:PER_ID_NO_CNTRB_BY'),
         p_arg_values => webdb.wwv_standard_util.string_to_table2 (
             p_batch||':'||upper(p_empno) ));
end home_call_contrib_form_pr;
```

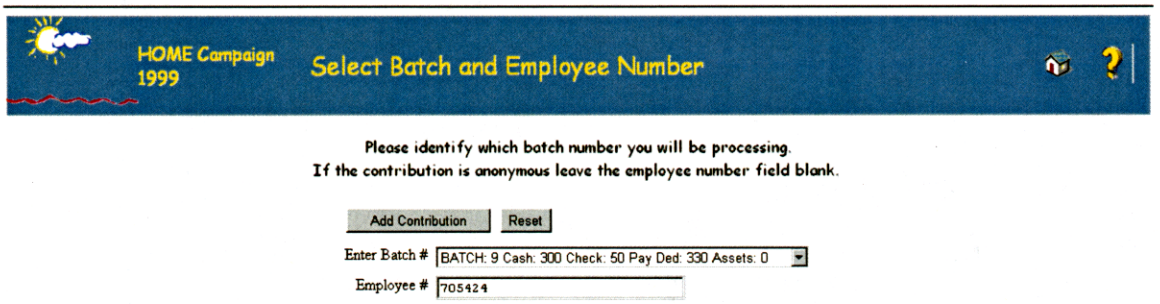



Figure 3. WebDB form built on procedure above serving as the Master.

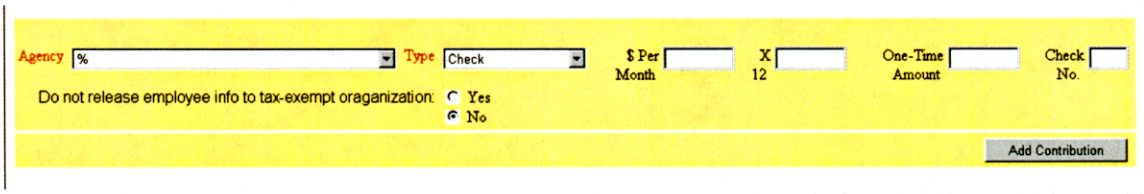



Figure 4. WebDB form called by procedure above serving as the Detail. This form is built on the base table HOME_CNTRB_SENS. Note: Batch number and employee id are hidden fields.

Note: Passing parameters into the WebDB PL/SQL table shown above, provides the ability to retrieve those same values later during processing. Below is the example to retrieve the HOME_BATCH_ID_NO and assign it to a variable. Remember that this table is only available in memory for this piece of PL/SQL. Once you hit the 'END;' of your PL/SQL block the information contained within is no longer addressable.

```
x varchar2(100);
x := WEBDB.wwv_name_value.get_string(
                                        l_arg_names,
                                        l_arg_values,
                                        'HOME_BATCH_ID_NO');
```

## Lesson Three – Empower With Multiple Components

WebDB – "Advanced PL/SQL Tab"

Within the Form and Report Builder exists the "Advanced PL/SQL Tab". This tab provides the ability to significantly add value to the functionality and flexibility of your applications. A good example was the requirement to have multiple data elements exist on one form. Using the "Advanced PL/SQL" option, we were able to display a variety of data using stored procedures, other WebDb components, and display buttons based on a user's privilege in order to enhance the form.

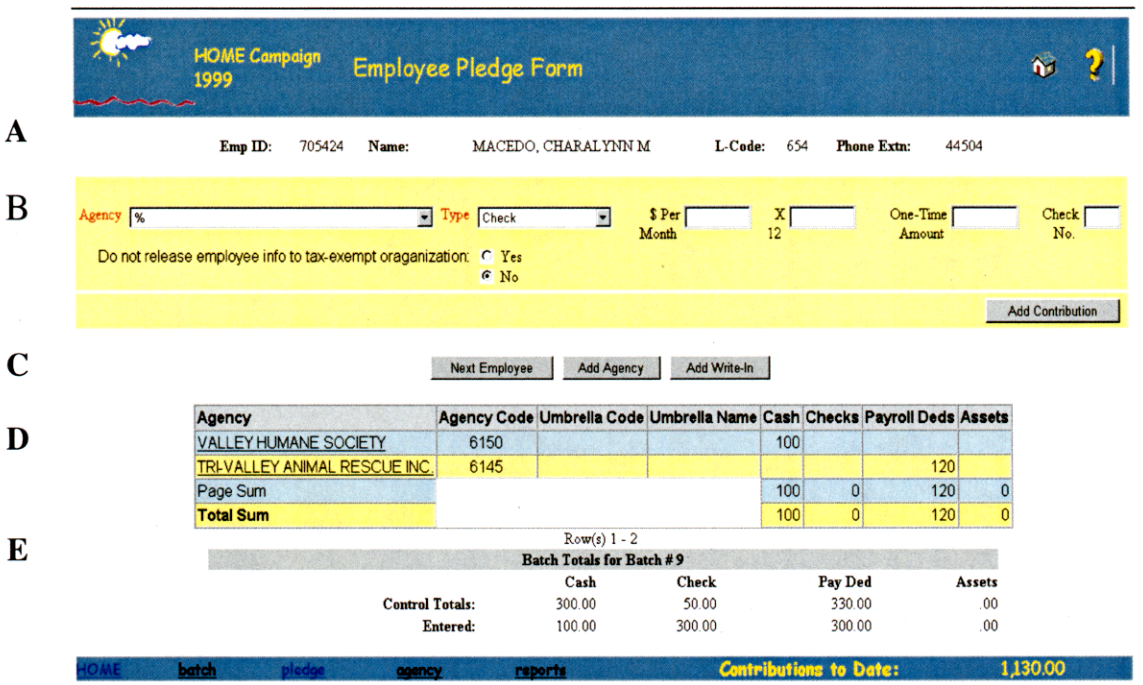



Figure 5. WebDB Form with multiple components

Taking a look at the "Advanced PL/SQL Tab" you can match the cells with the sections of the form above. You can see that the form above is actually 6 different components in all. Section 'B' is the only section germane to this base WebDB form.

Each cell shown below will accept PL/SQL code that runs at different points during the execution of the HTML code generated by WebDB. It is flexible enough to embed new code, call existing stored source, or call existing WebDB components.

A. PL/SQL Procedure

C. Buttons
D. WebDB Report
E. PL/SQL Procedure

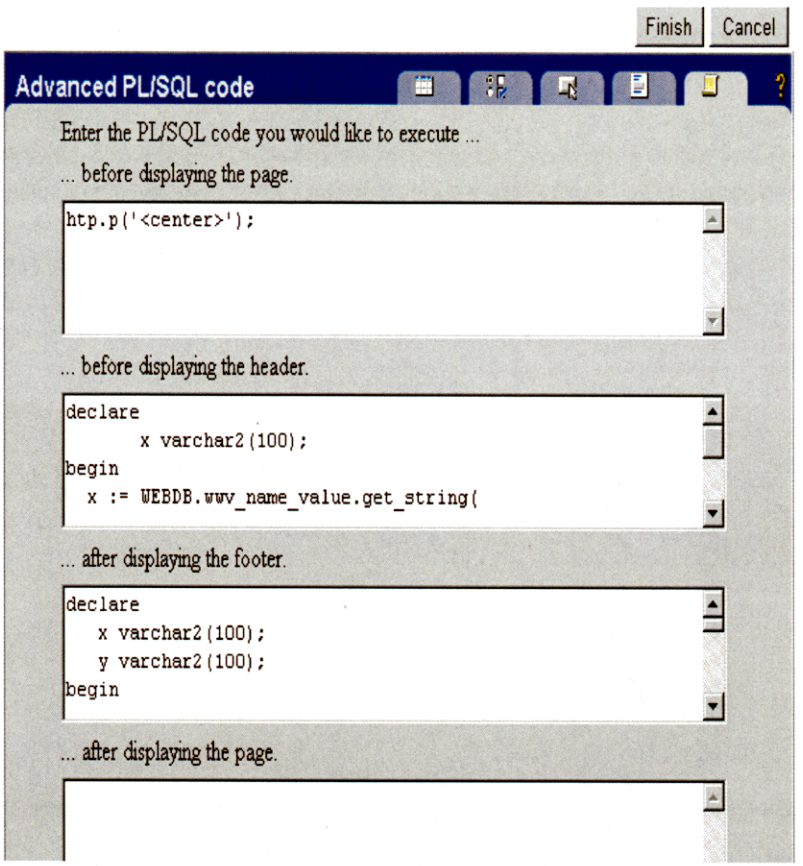



Figure 6. Advanced PL/SQL Tab within the Forms and Reports Component Builders

Note: When using multiple components on one form we encounter a scenario where the same WebDB component is called multiple times to accept multiple contributions. This became a nightmare during compiling final code into production. The contribution form (detail) above is called by the master which is built on stored source. In order for this to compile, the package spec of the master WebDB component must exist before the detail WebDB component can compile. Lastly, the stored source that the master WebDB component was built from must be compiled. Are you confused? This set up can be a major drawback and is not recommended when using this many components.

## Lesson Four – Real Time Data:

WebDB - The <ORACLE> Tag

Oracle WebDB introduces the concept of an <ORACLE> html tag. This tag allows the developer to build a dynamic flow of information into a static html environment. Between the open and close tags the developer can build a SQL query or PL/SQL block that executes in the database to retrieve dynamic data and display it. This tab type effectively operates the same as server-side includes of the old .shtml model.

In the HOME application we chose to display a "real time" running total in the footer of the template which was displayed on every page. Every time the html page is painted, the code embedded between the <ORACLE> tag is executed. Another good use of this tag in our application was the restriction of links to a user by privilege. A simple procedure was written to validate the user's db role and painted the hypertext links accordingly.


Figure 7. Footer with dynamically created links and real time data.

Note: Be cautious of performance when embedding a PL/SQL block within the <ORACLE> tag. We ran into a few performance problems when using multiple tags within one template.

## Lesson Five – Reusability:

WebDB – Shared Components

Oracle WebDB offers a variety of shared components that satisfy the need for "Reusability". The most common elements such as List of Values (LOV), Links, and Interface Templates are great time savers. Build once and use or re-use multiple times!

Interface Templates can be a flexible enhancement to any application by using more than one. One possibility is to build menus on one template and content pages on another. This allows you to customize the header, footer, and background options throughout your application, while at the same time providing a common look and feel. Our requirement was to have one standard template with background graphics for all pages that contained menus, and another standard template that contained the application forms and reports.

Another good example of multiple templates is the ability to create and use a blank template. We received a request to display a graphical chart on the same page as a tabular report. Notice in Figure 8 below that the bar chart was built using the WebDB Chart Wizard and placing it on a blank template. This provides the ability to emulate a stacked environment which is transparent to the user.

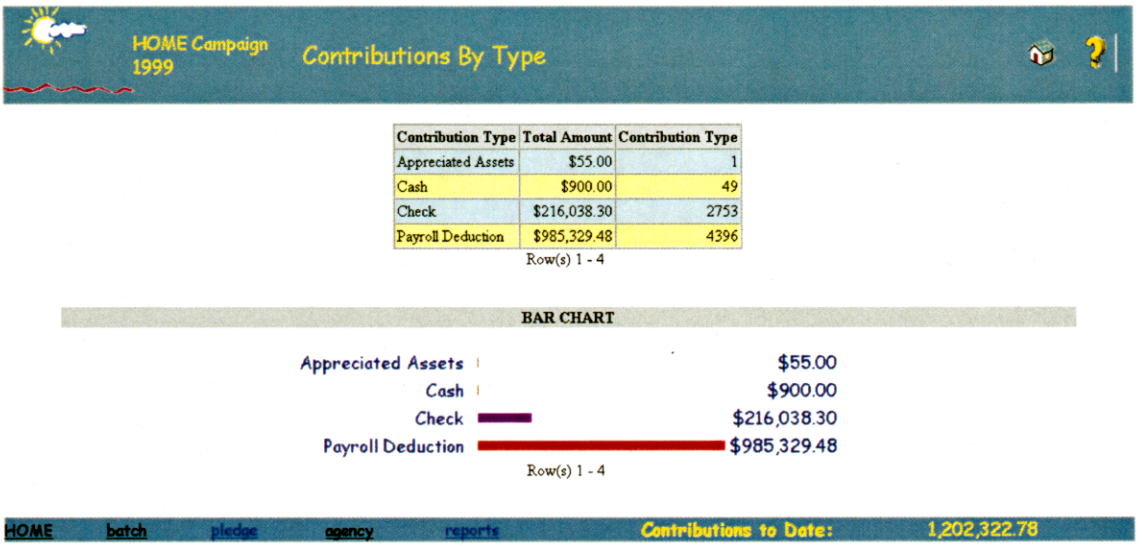



Figure 8. Example of blank template when viewing multiple components on the same page.

List of Values (LOV) is another re-usable feature in WebDB. A LOV can be built on a static list of values or a SQL statement. The bonus here is the display format options such as combo box, radio buttons, multiple selects, check boxes, etc.. One draw back is that WebDB only allows for one column to be displayed and one column to be passed as the value. Our requirement was to display the agency name and the corresponding United Way agency code since there may be two agencies with very similar names.

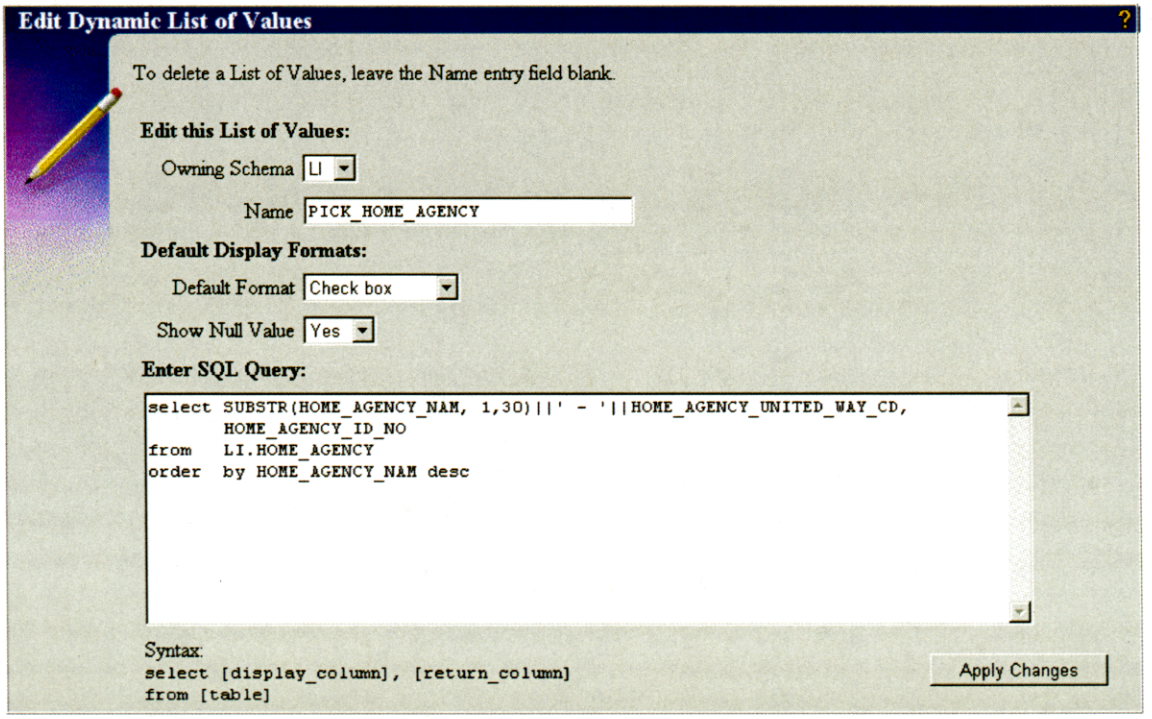



Figure 9. Example of a Dynamic LOV displaying two data elements.

## Conclusion

The main lesson to be learned is that the WebDB Component Builder in it's own right is a very useful tool. The easy-to-use wizardry of WebDB and the sophistication that custom PL/SQL code can offer, can become a very powerful way to enhance application development. There is never just one way to approach or solve a problem. Don't let yourself be confined to the tool itself. Breaking out of the box can be a very positive thing!

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.