Customizing Access Parameter Queries

[Revised and Updated 15 August 2018]

Everyone likes parameter queries! The database developer doesn't have to anticipate the user's every requirement, and the user can vary their enquiries without having to get involved with query design. But there's one question I'm always asked by my Access students. Can you customize the parameter input dialog?

The basic input dialog box that appears when you run a parameter query simply asks the user for some text input. Here is a standard parameter input dialog (*Fig. 1*)...

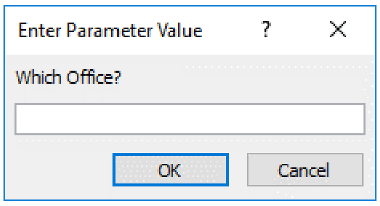


Fig. 1 A standard parameter dialog box.

It serves its purpose, but there is room for improvement. The title "Enter Parameter Value" is guaranteed to frighten the life out of a new user! Perhaps it could say something a little more friendly. Let's assume they've decided to continue and type something. What are they allowed to type? Perhaps they can't remember the names of all the offices or they might make a typo. Couldn't we add a combo box with a list of choices? The query might require several separate dialogs for different fields. Wouldn't it be useful if they could be combined into a single friendly dialog box?

So, can you customize the parameter input dialog? No, you can't. But you can do something much better. You can build a completely new one. In fact with this method you don't even build a parameter query. You build a regular query that takes its criteria from a special form. You have three things to do....

1. Build a dialog box with as many combo boxes as you need.
2. Design a query to read its criteria from the information on the dialog box.
3. Create a macro or visual basic procedure to tell them both what to do.

This tutorial explains all of these steps in detail so, if you are already an experienced user, you might want to skip straight to the bits that interest you.

Finally, there is a section illustrating some useful variations on this technique such as allowing for Null entries (when the user leaves the combo box empty).

I'm using a sample database containing details of the employees of a company that has offices in a number of different cities. Each office has several different departments. I'm going to build a query that enables the user so view an employee list selected by Office and Department. You can download a sample copy of the completed database file from www.fontstuff.com/access/files/acctut08.zip

Step 1: Build the Dialog Box

You are going to build a dialog box to replace the one that the user would normally see when running a parameter query. The dialog box is just a form built using the Access form design tools. It will contain two combo boxes, one for Office and one for Department, and a couple of command buttons to make things happen.

1.1 Create a New Form in Design View

Click the **Create** tab on the Access ribbon then click the **Form Design** button to create a new form in Design View. A form design tab opens containing a blank form. You don't need to specify a table or query for the form's data. This is going to be an "unbound" form, one that isn't linked to a data source. Your design window is probably maximized at this point but later you will tell Access to display the form as a dialog box.

Start by dragging the edges of the form to make a rectangle about 7 cm wide by 4 cm tall. It should look something like this (*Fig. 2*)...

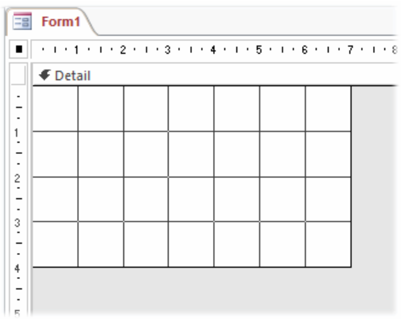


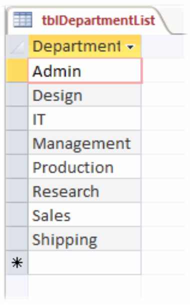
Fig. 2 Drag the form's background to the desired size.

Now you are ready to put some objects on the form. Access will have added a collection of Form Design Tools tabs to the ribbon. The tools necessary for this task are all located on the Design tab.

1.2 Prepare the List Data

I'm going to put two combo boxes on the form, one listing the Offices and another listing the Departments. But first, I need something that Access can use to build the lists. If you have ever used the wizard to create a combo box you will know that you have several choices. You can create a *table*, listing each item you want to appear on the list; you can build a *query* that makes the list from a data source such as one of the tables already in the database; or you can simply type your list on to the combo's properties sheet. Each method has its own advantages. A query, for example could be self-updating. A typed list would be easiest but fiddly to change later. I've chosen to create a couple of tables. Their data won't change often, but when it does it will be easy to update them.

I've called the tables *tblDepartmentList* and *tblOfficeList* and each has just one field, *Department* and *Office* respectively (*Fig. 3*).



tblDepartmentList

Department
Admin
Design
IT
Management
Production
Research
Sales
Shipping

Fig. 3 A table is used to populate the combo box list.

> TIP: Why the strange names for the tables? Access developers follow certain conventions when naming database objects. This becomes important when you get involved with VBA code where spaces in object names aren't allowed (and it's a pain typing all those underscores), and where prefixes such as "tbl" help to identify what sort of object is being referred to (e.g. tbl = table, qry = query, frm = form).

Once you have created your tables, you are ready to add the combo boxes to the form.

1.3 Add the Combo Boxes

First, switch off the *Use Control Wizards* option on the Design tab of the ribbon (*Fig. 4*). This will prevent the wizard from running when you select the Combo Box tool. You are going to do it the "hard" way! If you forget to disable the wizard, don't panic, just click the **Cancel** button on the first window of the wizard when it appears.

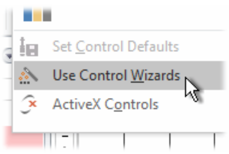


Fig. 4 Switch off the Control Wizards option.

To add a combo box click the **Combo Box** button (*Fig. 5*) on the toolbox then click on the form about 1 cm down and 3 cm in from the left.

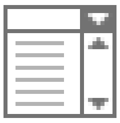


Fig. 5 The Combo Box tool.

Access creates an unbound combo box on the form, and a text label to go with it (*Fig. 6*).

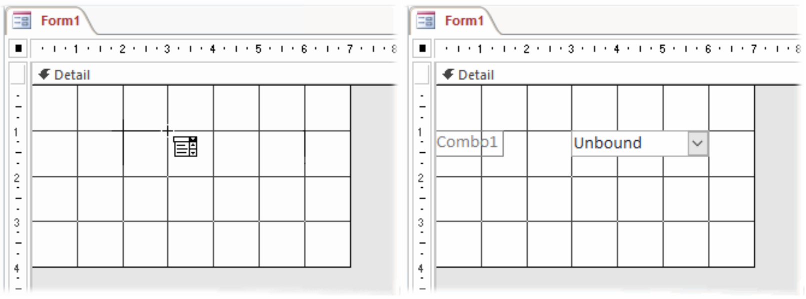


Fig. 6 Use the Combo Box tool to position and add a new combo box.

Click the label to select it. You can press the **[Delete]** key on your keyboard to delete the label, or click the label a second time to insert your cursor and edit the text to something meaningful. You can resize or move the control (an object on a form such as a combo box is called a "control") or its label by dragging the dots that appear around the edge of the object when it is selected. Watch the cursor to see what will happen when you drag it (*Fig. 7*)...

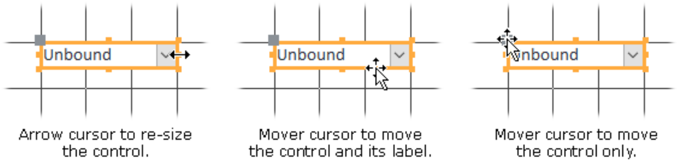

Unbound
Arrow cursor to re-size the control.
Unbound
Mover cursor to move the control and its label.
Unbound
Mover cursor to move the control only.

Fig. 7 The cursor indicates its function when dragging.

TIP: Here's an easy way to re-size a label to fit its text. Just double-click one of the small dots and the label will snap to the correct size.

TIP: It can be really fiddly moving and resizing those labels and controls with the mouse. I prefer to use the keyboard... Select the object then, on the keyboard, use **[Arrow]** to move or **[Shift]+[Arrow]** to re-size. Hold the **[Control]** key down at the same time to fine tune the movements. The up, down, left and right arrow keys all do different things so experiment!

Now you have an "unbound" combo box on your form. The term unbound indicates that it is not connected to any field. It will just display a value that we can make use of later. In this example I need two of these so I'll repeat the process (*Fig. 8*).

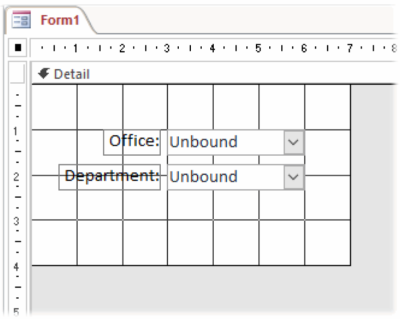


Fig. 8 A second combo box is added to the form.

TIP: If you have an object like a control or command button on your form and you want another exactly the same, select the object then **Copy** and **Paste** (don't bother to click anywhere in between) and you've got an exact copy of the original. A quick way to do this is to select the control you want to duplicate then hold down the **[Control]** key and press **C** then **V**. An exact copy of the control will appear directly below it.

1.4 Set the Combo Box Properties

You may be tempted to switch to Form View and check out your combo boxes, but they won't work yet. We have to give them their instructions by setting their *properties*. Select the combo box whose properties you want to set and click the *Properties* button (*Fig. 9*) on the Design tab of the ribbon (or right-click the combo box and choose *Properties* from the context menu).

Fig. 9 The Properties button.

This opens the Properties pane at the right-hand side of the Form Design window.

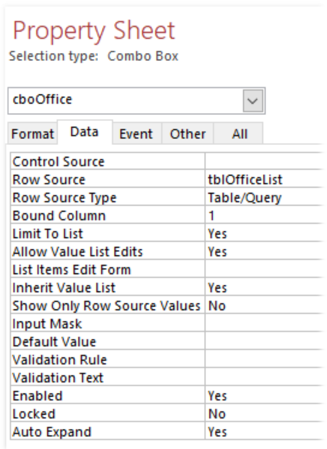
The first thing to do is give the combo box a meaningful name (an object's "name" is a different thing from its "label"). Click the "Other" tab on the properties pane. You'll see the name that Access gave it, something like *Combo1*, but it makes sense to change it to something meaningful so you don't get confused when you have to refer to it in your VBA code, or in a macro or query. I've called mine *cboOffice* (note the propellerhead naming convention again).

Next we have to tell the combo box where to get its list. Switch to the "Data" tab of the Properties pane. If you used a table or query to make your list (I used tables) leave **Row Source Type** set to **Table/Query**. Then click in the box next to **Row Source**, click the down-arrow and choose your table or query from the list.

If you chose to type out your list instead, change the Row Source Type to **Value List** then, in Row Source, type the items you want to see in your list, separated by semicolons (**;**). Like this...

"Birmingham"; "London"; "Manchester"; "New York"; "Paris"

The are several more properties that you can change if you wish (*Fig. 10*). I've chosen to set the **Limit To List** property to **Yes** - this prevents the user from typing something that isn't on the list. I've also left the **Auto Expand** setting as **Yes** - this means that the user can type the first one or two letters of an entry for Access to fill out the full name automatically. Auto Expand is either really useful or profoundly irritating depending on what's in your list, so you can turn it off if you want.


Property Sheet
Selection type: Combo Box

cboOffice

Control Source	
Row Source	tblOfficeList
Row Source Type	Table/Query
Bound Column	1
Limit To List	Yes
Allow Value List Edits	Yes
List Items Edit Form	
Inherit Value List	Yes
Show Only Row Source Values	No
Input Mask	
Default Value	
Validation Rule	
Validation Text	
Enabled	Yes
Locked	No
Auto Expand	Yes

Fig. 10 Set the properties of the combo box.

I haven't bothered to set a Default Value, because I want the user to have the option of leaving the entry blank. I could have set a Default Value of **"London"** for example (remember to put quote marks around text).

TIP: To make things easy for your users the items in your combo box list should be in alphabetical order. If your list is based directly on a table the order can be disrupted if new items are added. For this reason I usually use a query as the row source, even when working with a simple one-column table, because this allows me to sort the list items so that they always appear in the correct order. There is no need to create a separate stored query for this. Instead of selecting a table as the Row Source click in the Row Source textbox then click the **Build** button **[...]** to open the query builder where you can create an embedded query that sorts the table data for you. When you close the query builder it generates a SQL statement for the Row Source property.

Other helpful properties are **Status Bar Text** and **Control Tip Text** which you'll find on the "Other" tab. They both offer help for the user and both can accept up to 255 characters of text (make sure your status bar is long enough for what you type!). Status Bar Text appears on the status bar at the bottom of the Access window when the user enters the combo box. Control Tip Text appears in a pop-up box next to the mouse pointer when the user points at the combo box. If you need specific help on any of the properties just click on the appropriate property and press the **[F1]** key on your keyboard.

Set the properties for each of the combo boxes on your form.

1.5 Draw the Command Buttons

I'm using two command buttons. One for **[OK]** that will run the query and close the form when the user clicks it, and one for **[Cancel]** that will just close the form if the user changes their mind.

Click the **Command Button** button on the Design tab of the ribbon, then click on the form where you want your button to appear. You may want to resize the button. Do it the same way as for the combo boxes. Create a second button (copy/paste is quick and easy) and arrange them to suit yourself (*Fig. 11*).

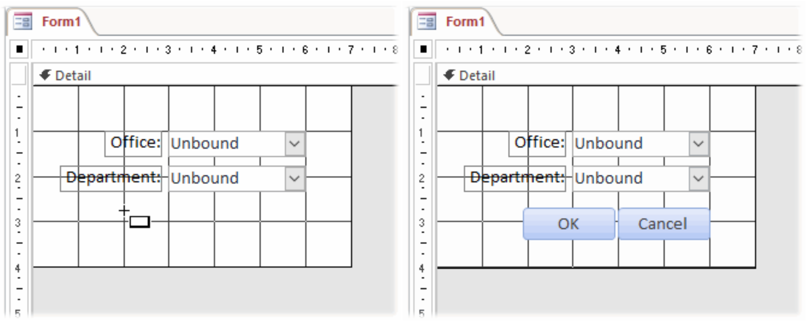


Fig. 11 Use the Command Button tool to add two command buttons to the form.

1.6 Set the Command Button Properties

Give your buttons sensible captions like **OK** and **Cancel**. Use the properties window to set their properties as follows...

OK Button: Caption: *OK*, Name: *cmdOK*, Default: *Yes*
Cancel Button: Caption: *Cancel*, Name: *cmdCancel*, Cancel: *Yes*

Setting the *Default* property to *Yes* for the **OK** button makes it the form's default button. As long as the user hasn't selected another button on the form (by tabbing to it for example), pressing the **[Enter]** key on the keyboard will have the effect of pressing the button. You may *not* want to set this property if you think the user might press enter by mistake (users do that sort of thing!). Similarly, setting the *Cancel* property *Yes* for the **Cancel** button means that button will be pressed if the user presses the **[Esc]** key on the keyboard.

We're going to use the **OK** button to run the query (so pressing **[Enter]** after selecting from the combos makes sense), and the **Cancel** button will close the form without running the query (users expect things to go away if they press the **[Esc]** key). These properties simply assign actions to the key presses - we still have to create procedures to tell the buttons what to do.

At this point you might like to decorate the background of the form to suit the colour scheme of your database (*Fig. 12*). To change the background colour, click on the Detail area of the form and change its *BackColor* property.

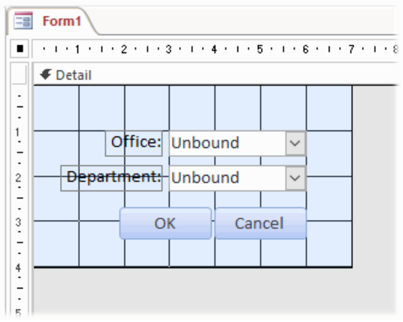


Fig. 12 Make the form's background a suitable colour.

One property we haven't set yet is the *On Click* event property. I'll return to that later. First we have to tidy up the form and make it look like a proper dialog box...

1.7 Turning the Form into a Dialog Box

To make our form look more professional we need to change some of its properties so that it looks like a proper dialog box. An Access form has several features that a dialog box does not need, like record selectors and navigation buttons. Double-click the Form Selector button (it's the small rectangle located where the two rulers meet) to select the form itself (*Fig. 13*) and display the form properties window. When the form is selected a black dot appears in the centre of the Form Selector.

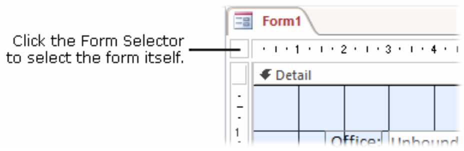


Fig. 13 Select the form itself.

Go to the **Format** tab of the Form Properties pane and set the properties as follows:

Caption: *[Choose a suitable title]*, Scrollbars: *Neither*, Record Selectors: *No*, Navigation Buttons: *No*, Dividing Lines: *No*, Auto Center: *Yes [Optional]*, Border Style: *Dialog*, Control Box: *No [Optional]*.

Setting the Border Style to *Dialog* automatically removes the Min/Max buttons from the form. You can also choose to set the Close Button property to *No* (which disables it rather than removing it), or to set the Control Box property to *No* (which removes both the Control Box* and the Close Button). I have chosen the latter option in this example but sometimes I prefer not to do this. You must give your user some sort of 'escape route'. They might have arrived at this dialog box by mistake and just want to get rid of it. Here, our custom Close button will let them do that. Users will often close a form or dialog using its own close button (the one in the upper right corner marked with a cross **[X]**). You may not want them to do this if, for example, the dialog is shown mid-way through a procedure. Unexpectedly closing a dialog might crash your code! So when I want complete control over the user's actions I provide my own close button that I can program events the way I want.

> *NOTE: What! You never heard of the Control Box? It's the tiny icon in the upper left corner of almost every window. Click on it with your mouse to get a menu of window options.

Go to the **Other** tab of the Form Properties pane and set the following property to make sure that the form appears as a dialog and not as a tabbed form or window:

Pop Up: *Yes*

Take a look at your form dialog box in *From View* and if necessary resize the detail area so that it is a suitable size for what it shows. You may want to re-arrange the contents so that everything looks good. Here is our (almost) finished dialog box (*Fig. 14*):

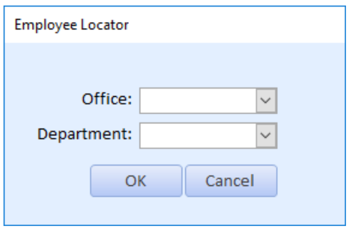


Fig. 14 The form’s design is complete.

It's time to close and save the form as we are about to turn our attention to the query. I have called this form *frmEmployeeLocator*. You can, of course, call it anything you like.

The next step is to design the query that will make use of the choices that the user makes from the form's combo boxes...

Step 2: Design the Query

2.1 Create the Query

Select the Create tab of the Access ribbon and click the Query Design button to create a new query in Design View. Build your query as you would normally, choosing the table whose data you want to query and adding the fields you want to see. Don't add any criteria yet. In my example, I'm going to set the criteria for the *Office* field of my Staff table to the value that the user chose on the *cboOffice* combo box on my new custom dialog box. I'll do the same for the *Department* field.

2.2 Set the Field Criteria

We need to tell the query to use as its criteria the values currently showing in the appropriate combo box of the custom dialog box. If you know what to type you can enter the instructions directly into the criteria cell. I find it easier to use the **Build** tool. Right-click in the first criteria cell of your chosen field (here I'm using the *Office* field) and choose **Build** from the context menu (*Fig. 15*).

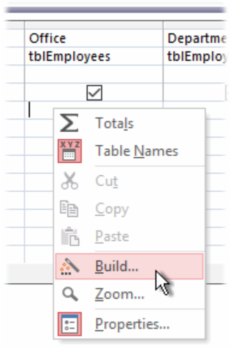
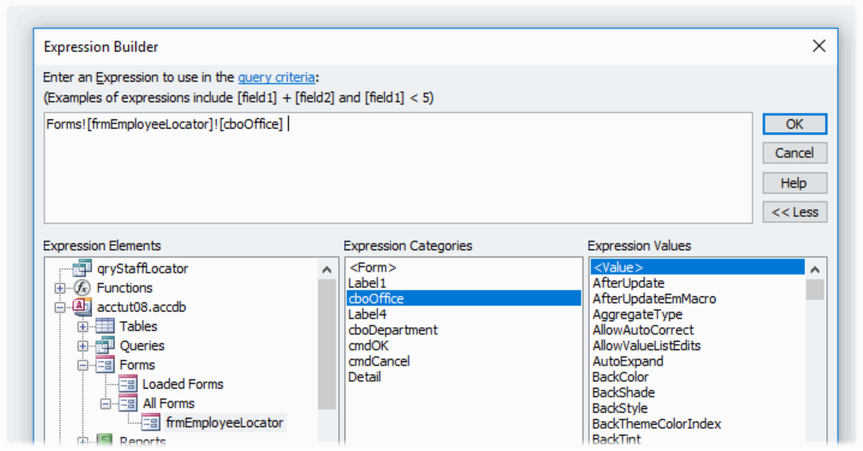


Fig. 15 Select the Build tool.

This opens the **Expression Builder** window. In the lower part of the window there are three columns. In the left column click the [+] next to the name of your database to expand its contents. Then expand *Forms* and *All Forms* until you can see the name of the dialog box you just created (mine was named *frmEmployeeLocator*) and click it.

This displays in the centre column a list of all the objects (controls, labels etc.) on this form. Find the name of the combo box that contains the list appropriate to the field whose criteria you are setting (mine is *cboOffice*) and double-click it. Its full designation appears in the upper section of the Expression Builder (*Fig. 16*).


Expression Builder

Enter an Expression to use in the query criteria:
(Examples of expressions include [field1] + [field2] and [field1] < 5)

Forms![frmEmployeeLocator]![cboOffice]

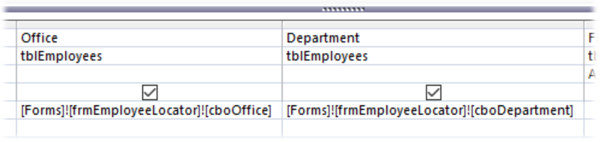
Expression Elements: qryStaffLocator, Functions, acctut08.accdb, Tables, Queries, Forms, Loaded Forms, All Forms, frmEmployeeLocator, Reports

Expression Categories: <Form>, Label1, cboOffice, Label4, cboDepartment, cmdOK, cmdCancel, Detail

Expression Values: <Value>, AfterUpdate, AfterUpdateEmMacro, AggregateType, AllowAutoCorrect, AllowValueListEdits, AutoExpand, BackColor, BackShade, BackStyle, BackThemeColorIndex, BackTint

Fig. 16 Select the new dialog in the Expression Buider.

Click the **[OK]** button to close the Expression Builder and return to the query design window, where you will see that the correct information has been placed into the criteria cell. Repeat this for each of the combo boxes (taking care to apply the correct combo box criteria to each field). The result will look something like this (*Fig. 17*)...



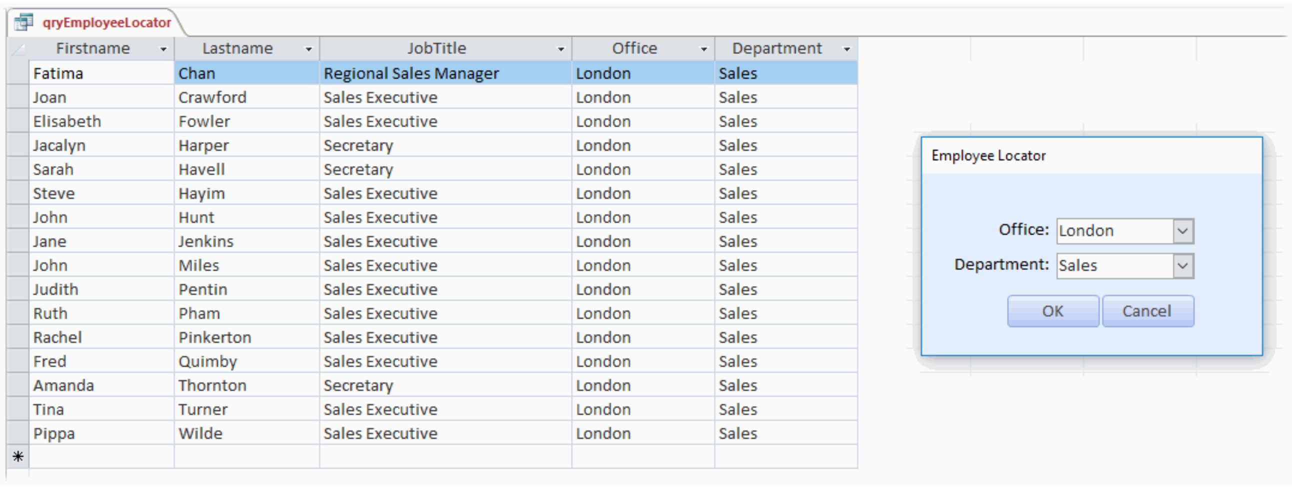
Office	Department
tblEmployees	tblEmployees
☑	☑
[Forms]![frmEmployeeLocator]![cboOffice]	[Forms]![frmEmployeeLocator]![cboDepartment]

Fig. 17 The query contains a reference to each combo box.

We haven't finished yet, but you might like to try out the query at this point...

2.3 Test the Query

Leave the query window open but go to the Navigation Pane and open your custom dialog box. Make selections from the combo boxes but don't close the dialog box. Now switch back to the query design window and run the query by pressing the **Run** button on the Query Design tab of the ribbon. If everything has been done correctly, your query will proceed using the criteria you selected in the dialog box and you will see the appropriate result (*Fig. 18*).



qryEmployeeLocator

Firstname	Lastname	JobTitle	Office	Department
Fatima	Chan	Regional Sales Manager	London	Sales
Joan	Crawford	Sales Executive	London	Sales
Elisabeth	Fowler	Sales Executive	London	Sales
Jacalyn	Harper	Secretary	London	Sales
Sarah	Havell	Secretary	London	Sales
Steve	Hayim	Sales Executive	London	Sales
John	Hunt	Sales Executive	London	Sales
Jane	Jenkins	Sales Executive	London	Sales
John	Miles	Sales Executive	London	Sales
Judith	Pentin	Sales Executive	London	Sales
Ruth	Pham	Sales Executive	London	Sales
Rachel	Pinkerton	Sales Executive	London	Sales
Fred	Quimby	Sales Executive	London	Sales
Amanda	Thornton	Secretary	London	Sales
Tina	Turner	Sales Executive	London	Sales
Pippa	Wilde	Sales Executive	London	Sales

Employee Locator

Office: London

Department: Sales

Fig. 18 Test the new dialog box.

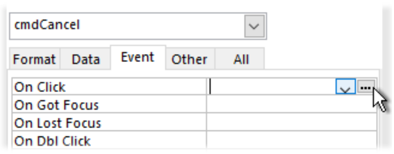
If you need to modify the query in any way do it now, then close and save it, giving it a suitable name (mine is called *qryEmployeeLocator*). We are now ready to create the instructions that will link the query with the dialog box...

Step 3: Create the Code

The final task is to create the instructions that operate when the user clicks the **OK** or **Cancel** buttons on the dialog box. You could do this using an Access Macro but I have chosen to use VBA code because this gives me much greater control. It is also an easy introduction to writing VBA code if you haven't done it before. Open your custom dialog box in Design View if it is already open right-click on it and choose *Design View* from the menu.

3.1 Programming the Cancel Button

The easy one first! The Cancel button needs to be programmed to close the form without running the query when it is clicked by the user. Right-click the Cancel button and choose **Properties** from the context menu to open the Properties Window. Switch to the **Event** tab. Here you will see a list of events associated with the Cancel button. We are going to attach a VBA procedure to one of these.



cmdCancel

Fig. 19 Create an Event Procedure for the Cancel button.

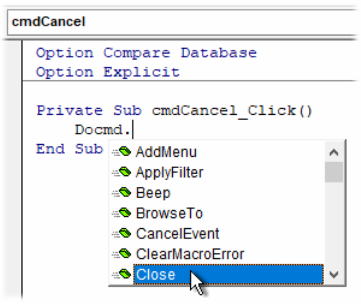
Click in the box next to **On Click**, then click the **Build** button (the one with the **[...]** symbol) (*Fig. 19*). Choose **Code Builder** from the dialog box that appears then click **[OK]** to open the Visual Basic Editor. The first parts of the code are already in place (*Listing 1*):

Listing 1:

```
Private Sub cmdCancel_Click()

End Sub
```

Place your cursor in the empty line between the *Sub* and *End Sub* statements and press **[Tab]** to indent your code (this makes it easier to read). Type *DoCmd* followed by a dot. When you type the dot a list of possible entries appears...

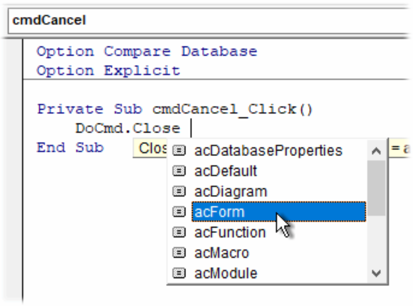


```
cmdCancel
Option Compare Database
Option Explicit

Private Sub cmdCancel_Click()
    Docmd.
End Sub
```

Fig. 20 Start writing the code to close the dialog box.

Choose *Close* from the list (*Fig. 20*) and type a space. Typing the space both enters the text *Close* into your code and also enters the space. This prompts the next list of possible entries to open...

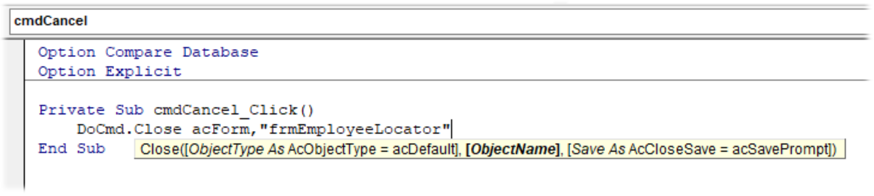


```
cmdCancel
Option Compare Database
Option Explicit

Private Sub cmdCancel_Click()
    DoCmd.Close
End Sub
```

Fig. 21 Continue writing the code to close the dialog box.

Choose *acForm* (*Fig. 21*) and type a comma (**,**). Typing the comma enters the text *acForm* into your code and also enters the comma. Now you can see a panel indicating the various pieces of information (called *arguments*) that this statement needs (*Fig. 22*). The current one is in bold, telling us that we should enter the form's name next. The remaining arguments are in italic, indicating that they are optional. Type, in quotes, the name that you gave your custom dialog box (mine is called *frmStaffLocator*).



```
cmdCancel
Option Compare Database
Option Explicit

Private Sub cmdCancel_Click()
    DoCmd.Close acForm,"frmEmployeeLocator"
End Sub
```

Close([*ObjectType As* AcObjectType = acDefault], **[*ObjectName*]**, [*Save As* AcCloseSave = acSavePrompt])

Fig. 22 The Visual Basic Editor tells you what it requires.

Your finished code module looks like this (*Listing 2*):

Listing 2:

```
Private Sub CmdCancel_Click()
   DoCmd.Close acForm, "frmStaffLocator"
End Sub
```

Close the code module window. The **On Click** property of the Cancel button now shows *[Event Procedure]* indicating that there is some code associated with that event. You can edit or view the code at any time simply by clicking the **Build** button.

TIP: If you want to remove the code associated with a particular event, just delete the entry *[Event Procedure]* from that event in the Properties Window.

Your code will be saved when you save the changes to the form so do that now. Switch the form to Form View and test your Cancel button. When you click the button the dialog box closes. Because you set the *Cancel* property of the Cancel button to *Yes*, pressing the **[Esc]** on your keyboard also closes the dialog box.

3.2 Programming the OK Button

The OK button needs to be programmed to run the query and then close the dialog box. The query already knows that it must get its criteria from the dialog box. I am getting the procedure to close the dialog box to deter the user from trying to run the query again whilst it is already open.

Use the same method as for the Cancel button to add an event procedure to the **On Click** event of the OK button. Your code should look something like this (*Listing 3*):

Listing 3:

```
Private Sub cmdOK_Click()
   DoCmd.OpenQuery "qryStaffLocator", acViewNormal, acEdit
   DoCmd.Close acForm, "frmStaffLocator"
End Sub
```

It is important that the code runs the query *before* it closes the dialog box. If the instructions were reversed the query would not be able to retrieve its criteria and the user would see an error message.

Note also that when opening the query you have a number of choices. You could, for example, choose to open the query *Read Only* (preventing the user from making changes to the data), or *Add* (allowing the user to add new entries but not edit or delete existing ones).

Save and test your custom dialog box. Remember that this kind of query has to be run from the dialog box. If you try to open the query by itself there will be an error (unless the dialog box is already open and showing choices). To do this simply place a command button on a form and add the appropriate code to its *On Click* event (*Listing 4*). For example:

Listing 4:

```
Private Sub cmdEmployeeLocator_Click()
   DoCmd.OpenForm "frmEmployeeLocator"
End Sub
```

4. Variations on a Theme

In the tutorial I built a dialog box with multiple combo boxes.

If the user leaves either combo box blank, the query will return no data. There are two ways to handle this.

1. You can insist that the user makes choices from all the combo boxes, by changing the code to check that both combo boxes contain a value (i.e. the user has chosen something). How...

2. You can modify the query to interpret an empty combo box as the user's wish to see *all* results for that particular field (i.e. leaving the *Department* combo box blank means you want to see records for all departments).

4.1 A Simple Single-Combo Dialog

If you need only one you can save the user the effort of clicking buttons. This demonstrates the technique at its simplest and most elegant. It looks just like a parameter query dialog but it has a combo box (*Fig. 23*).

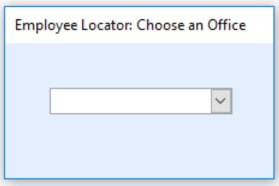
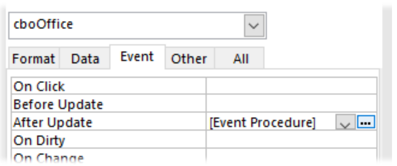


Fig. 23 A single-parameter dialog box.

If you are querying on a single variable field, then you need only one combo box. You don't need any buttons because you can run the query code when the user chooses an item from the list. You do this by attaching the event procedure to the *After Update* event of the combo box itself (*Fig. 24*)...


cboOffice

Format | Data | Event | Other | All

On Click	
Before Update	
After Update	[Event Procedure]
On Dirty	
On Change	

Fig. 24 Create an After Update event procedure.

The code looks the same (*Listing 5:*) it's just in a different place...

Listing 5:

```
Private Sub cboOffice_AfterUpdate()
   DoCmd.OpenQuery "qryEmployeeLocator", acViewNormal, acEdit
   DoCmd.Close acForm, "frmEmployeeLocator"
End Sub
```

I have not allowed for the user wanting to close the dialog box without running the query, and if I was feeling benevolent I might change the form properties to reinstate the control box/close button.

4.2 Refusing Null Entries

If you decide that the user *must* make a choice (i.e. they are not allowed to leave one of the combo boxes blank) you need to be able to check their entries before running the query. To do this, modify the code that runs when the user clicks the OK button to look something like this (*Listing 6*):

Listing 6:

```
Private Sub cmdOK_Click()
   If IsNull(cboOffice) Or IsNull(cboDepartment) Then
      MsgBox "You must choose both Office and Department."
      Exit Sub
   End If
   DoCmd.OpenQuery "qryEmployeeLocator", acViewNormal, acEdit
   DoCmd.Close acForm, "frmEmployeeLocator"
End Sub
```

Or for a more professional looking message box...

```
Private Sub cmdOK_Click()
   If IsNull(cboOffice) Or IsNull(cboDepartment) Then
      MsgBox "You must choose both Office and Department." _
         & vbCrLf & "Please try again.", vbExclamation, _
         "More information required."
      Exit Sub
   End If
   DoCmd.OpenQuery "qryEmployeeLocator", acViewNormal, acEdit
   DoCmd.Close acForm, "frmEmployeeLocator"
End Sub
```

The code has been modified to include an IF statement that checks to see if either combo box is empty. If so, a message is displayed and the procedure terminated. If not, the query runs as normal.

> TIP: When you modify existing code, Access sometimes displays an annoying habit of ignoring your changes! The solution is simple. In the code editing window select all your new code, **Cut** it (so that it disappears) then immediately **Paste** the code back again. It sounds crazy but it works!

4.3 Allowing Null Entries

Most people would assume that a blank combo box would mean they wanted to see everything. A query assumes the opposite so, if you leave any combo box empty the query returns no records. If you want to allow the user to leave any or all of the combo boxes empty you have to modify the query criteria. See also the tutorial *Parameter Queries: Handling Null Responses* at www.fontstuff.com/access/acctut07.htm.

For each query field that you want to allow a null entry modify your criteria from:

[Forms]![frmStaffLocator]![cboOffice]

to:

[Forms]![frmStaffLocator]![cboOffice] Or Like [Forms]![frmStaffLocator]![cboOffice] Is Null

When you do this and run the query for the first time you will find that Access has changed the way your criteria were written from the text above to something more complex. The criteria grid now contains several rows of entries. Leave it alone! Access has just broken down your **... Or Like ... Is Null** statement into its component parts, listing all possible combinations of null and not-null entries on separate lines. The query needs this but, thankfully, you can type it out the quick way!

> TIP: Sometimes you have a lot to type into a query criteria cell. It's important that everything is correct so make life easier for yourself! Right-click the cell and choose **Zoom** to open a large editing window that lets you see what you are typing.

Final Thoughts

Running your queries from a form or switchboard offers a high degree of user-friendliness and allows you to help the user make their choices by providing them with lists of options.

This means that the query has to be run from the form rather than from the query itself so you will probably want to build a switchboard listing available queries for the user to run.

Remember that, if you choose to allow the user to leave a combo box blank, the query definition starts to get complex. Limit your dialog boxes to two or three combo boxes, unless you are prepared to wait a long time for Access to execute a highly complex query!