

Microsoft Access 2007 – Forms: Combo Boxes

One of the most useful enhancements you can make to a basic Access form is to add some combo boxes (sometimes called drop-down or pull-down lists). They make data entry easier by providing a list of choices for the user. The user can open the list and search through the items or if they know what they want simply start to type and, if the list is arranged in alphabetical order, the combo box will jump to the nearest matching item. This has the advantage of making data entry both quick and accurate. You also have the option to limit the user's entry to items on the list, thus preventing them from entering anything else.

Adding a Combo Box to a Form

Start by opening your form in **Design View**. If you already have a text box on your form you can easily convert it to a combo box. Right-click the text box then choose **Change To** and **Combo Box**. If you use this method the combo box will already be linked to the same field as the text box.

Access provides a wizard to guide you through the steps of creating a combo box. It is activated automatically when you select the combo box tool in form design view. Whilst this is useful you have greater control when you work manually so these notes show you how to proceed without the help of the wizard.

If you are placing a combo box on your form as a new item, first switch off the **Control Wizards** feature by clicking the **Use Control Wizards** button if necessary so that it is not highlighted. These instructions explain how to make all the necessary settings manually so you won't need the help of a wizard. Select the **Combo Box** button in the **Controls** section of the **Form Design Tools – Design** tab of the ribbon then click on the form where you want the combo box to appear. Access places a new unbound (i.e. not yet linked to any field) combo box and an accompanying label on the form.

Once you have the combo box on the form you can position it accurately by dragging it with the mouse or by using the arrow keys on the keyboard. You can align it with existing controls on the form by making a multiple selection with the combo box and an existing control (**Click** on the combo box then **Shift+Click** on the other control to select both). Then use one of the **Control Alignment** buttons on the **Form Design Tools – Arrange** tab of the ribbon to line up the two controls.

If you want the combo box to be linked to a particular field (combo boxes are sometimes used for other purposes) select the combo box then click the **Property Sheet** button on the ribbon to display a list of the combo box properties. On its **Data** tab click on the **Control Source** property then choose a field from the drop-down list. You can also use the Property Sheet to give the combo box a meaningful name (usually the name of the field it is linked to).

Alternatively, click the **Add Existing Fields** button to open a field list then select the **Combo Box** button and, instead of clicking on the form, drag the appropriate field from the list to the form. This method creates a combo box that is already linked to the chosen field and whose label already has the correct caption.

Deciding What Goes on the List

There are three different ways to specify what items make up your combo box list: you can type out a list of items (called a *value list*), you can specify a table that contains the list of items, or you can create a query that generates the list. A combo box list can have one column or several depending on how it is being used. Multi-column lists are described later in this document.

Using a Value List

Use a value list when the list does not contain many items and when it is likely that the list items will not change in the future because if they do someone will have to change the combo box properties in design view. For example, a *Gender* field accepts only two possible entries: "F" and "M" (representing Female and Male respectively) so I provide a combo box to let the user know what they can enter and adjust its properties so they can't enter anything else (*Fig. 1*).

To add a value list to your combo box right-click the combo box and choose **Properties** to open the **Property Sheet** and select the **Data** tab. Set the **Row Source Type** property to **Value List** then type your list of items into the **Row Source** property text box, separating each item with a semi-colon (;). If you want to prevent the user from entering an item that isn't on the list you should also set the **Limit To List** property to **Yes** (*Fig. 2*). A new feature in Access 2007 is the

Allow Value List Edits property which allows users to add, change or delete items on a value list without getting involved with form design. If you don't want the user to be able to do this you should set this property to **No**.

Fig. 1 A simple list created by a Value List.

Fig. 2 Specifying a Value List in the Property Sheet.

Using a Table

Value lists become impractical when there are many items to display so if you list is a long one you can store it in a table. This also allows you to easily edit the list when necessary. Simple lists require only a single field such as the *Department* field in this example (Fig. 3).

Fig. 3 This combo box list is stored in a table.

To specify a table for your combo box list first create and save a suitable table (Fig. 4), set the **Row Source Type** property of the combo box to **Table/Query** then click on the **Row Source** property text box and choose your table from the list of tables and queries. Remember to set the **Limit To List** property to **Yes** if you want to prevent users from entering items that aren't on the list (Fig. 5).

tblDepartments	
Department	
Admin	
Design	
IT	
Management	
Production	
Research	
Sales	
Shipping	
*	

Fig. 5 Specifying a table as the Row Source.

Fig. 4 A single-field table stores the Departments list.

Using a Query

When the items on the list are likely to change it is usually best to create the list by using a query. This method can help to ensure that the list is always up-to-date. It is also a good way to make sure that items stored in a table, such as the table of *Departments* in the previous example, are always displayed in the correct alphabetical order when new items are added to the end of the table.

There are two ways to do this. You can create a query that generates the required list, save it, then specify it as described above (see: *Using a Table*). Alternatively you can create a query from within the form's design view. This method is very easy and does not require you to save a separate query in the database. Set the **Row Source Type** property of the combo box to **Table/Query** then click on the **Row Source** property text box and click the **Build** button ([...]) to open the **Query Builder**. This is the same tool that you use to create queries in Access. You use it to design and test your query but instead of saving the query in the database it copies the query's SQL statement into the **Row Source** property of your combo box.

In this example a list is being made from the *JobTitle* field of the *tblStaff* table, the same table that the form is based on. Since many people have the same job title, the query would normally show each person's entry and most of the job titles would be repeated many times in the list. To prevent this happening and ensure that each unique item appears only once use the query's **Property Sheet** to set the **Unique Values** property to **Yes**. It is usually appropriate to set the **Sort** order of the field to **Ascending** and you might also find it useful to enter **Is Not Null** into the criteria row to prevent an "empty" item from appearing in the list (reflecting any null values in the source data) (Fig. 6).

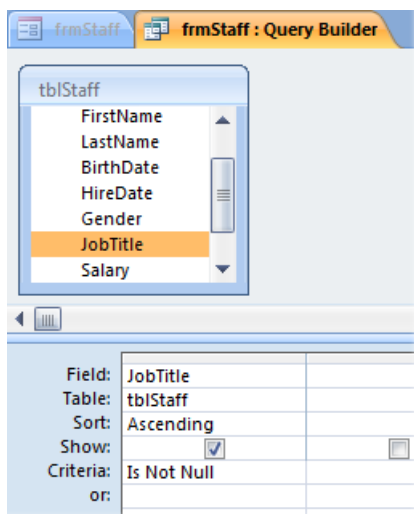


Fig. 6 Use the query builder to define the list.

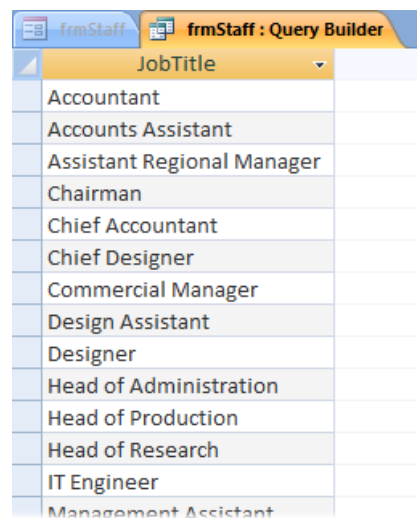
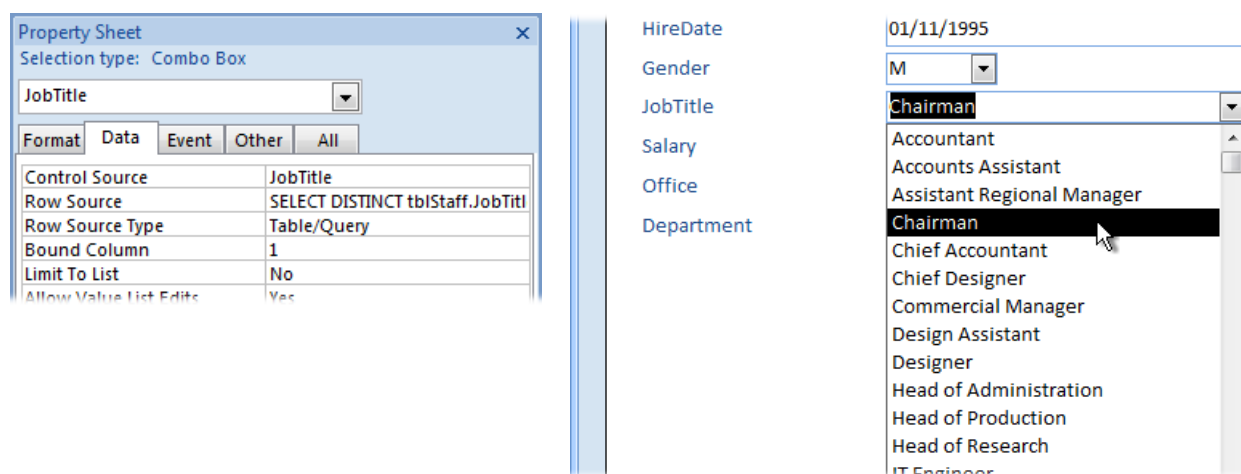


Fig. 7 The query generates a list of unique items.

Run the query in the usual way to check that the list is created correctly (Fig. 7) then finish by closing the query builder. Access will ask "Do you want to save the changes..." so answer **Yes**. You will see that the **Row Source** property now contains an SQL statement which will generate the combo box list each time the form opens.



Note: As in this example, it is possible to create a combo box list based on the same table as the form itself. You should set the **Limit To List** property of the combo box to **No** so that new entries can be made. These entries will then automatically appear on the list. However, any typos or spelling mistakes will also be added to the list! Since the list is only automatically refreshed when the form is opened, let the users know that they can update a form's lists at any time by clicking the **Refresh All** button on the **Home** tab of the Ribbon or by pressing the **[F9]** key on their keyboard.

Multi-Column Lists

Combo boxes can be a great help to the user, allowing them to work more quickly and accurately, but sometimes a single column of items isn't enough. Fortunately combo boxes can display more than one column. Creating a multi-column list involves several stages:

1. Prepare the data that will comprise the list and define it as the **Row Source**.
2. Use the **Column Count** property to specify how many columns the combo box has.
3. Use the **Bound Column** property to specify which column contains the data to be saved in the field.
4. Use the **Column Widths** property to specify how wide each column should be or whether any columns should be hidden.

Multi-Column Value Lists

You might want to add an additional column simply to provide the user with a little more information to help them make their choice. In the *Gender* example described earlier (see: *Using a Value List*) you might want to show that "F" means Female and "M" means Male.

As with a single-column value list, you set the **Row Source Type** to **Value List** and enter the list items into the **Row Source** property in the required order, separated by semi-colons. Work a row at a time, entering the item for the first column, then the item for second column and so on then do the same with the items for the next row and continue until all the data has been supplied.

For example, a two column *Gender* list might require you to enter: **F;Female;M;Male** In my example I want only the letters F or M to be entered into the field so, because these items are contained in the first column, I set the **Bound Column** property to **1** (Fig. 8). It is also necessary to tell Access how many columns should be displayed so I set the **Column Count** property to **2** (find it on the **Format** tab of the **Property Sheet**). You can let Access decide how wide to make each column but it is usually best to decide this yourself using the **Column Widths** property. Enter the required widths in centimetres, separated by semi-colons e.g. **0.5;1.5** (Access sometimes adjusts your entries to the nearest equivalent to fit its design grid e.g. *0.501 cm;1.501 cm*). The width of the list can be different from the width of the combo box. You can specify the total width of the list using the **List Width** property. I do this by adding the column widths together and, if there is going to be a vertical scrollbar because you have a long list, I add an extra 0.5 cm for this. If you don't supply a value for the **List Width** property and accept the default setting of **Auto** Access makes the list the same width as the combo box itself (Fig. 9).

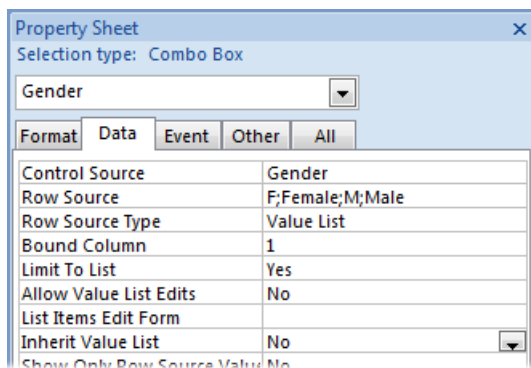


Fig. 8 Setting the Data properties (Row Source, Bound Column) of a Value List.

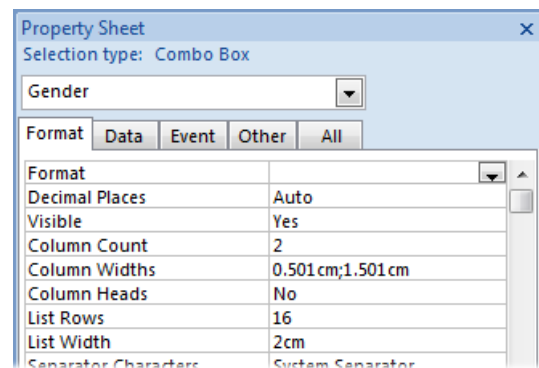


Fig. 9 Setting the Format properties (Column Count, List Width, List Width) of a Value List.

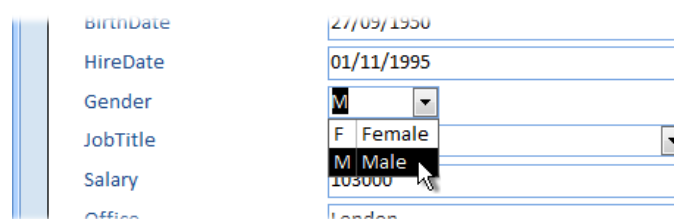


Fig. 10 The resulting two-column Value List.

Note: In Access 2007 if the list fails to display correctly make sure that the **Inherit Value List** property is set to **No**.

Multi-Column Lists from Tables or Queries

Additional columns are usually used to provide extra information for the user. This is particularly important when the actual value required in the underlying field is a code or number such as an AutoNumber. If you use AutoNumbers as your Primary key field and the user needs to enter, for example, a particular ProductID on an order or the StaffID of a SalesPerson they are unlikely to remember all the codes – you can add additional columns to the combo box to help them choose the right one.

Create your table or query is exactly the same way as for a single-column list (see: *Using a Table* and *Using a Query*) and include as many columns as you need, making the ID field the first column. (**Note:** although you can use either a table or a query as the **Row Source** a query offers much more flexibility and is usually the better method.) In this example (*Fig. 11*) the query creates six columns of data although only the first column will be entered into the form's **ProductID** field, the remaining five columns are included to help the user choose the correct item.

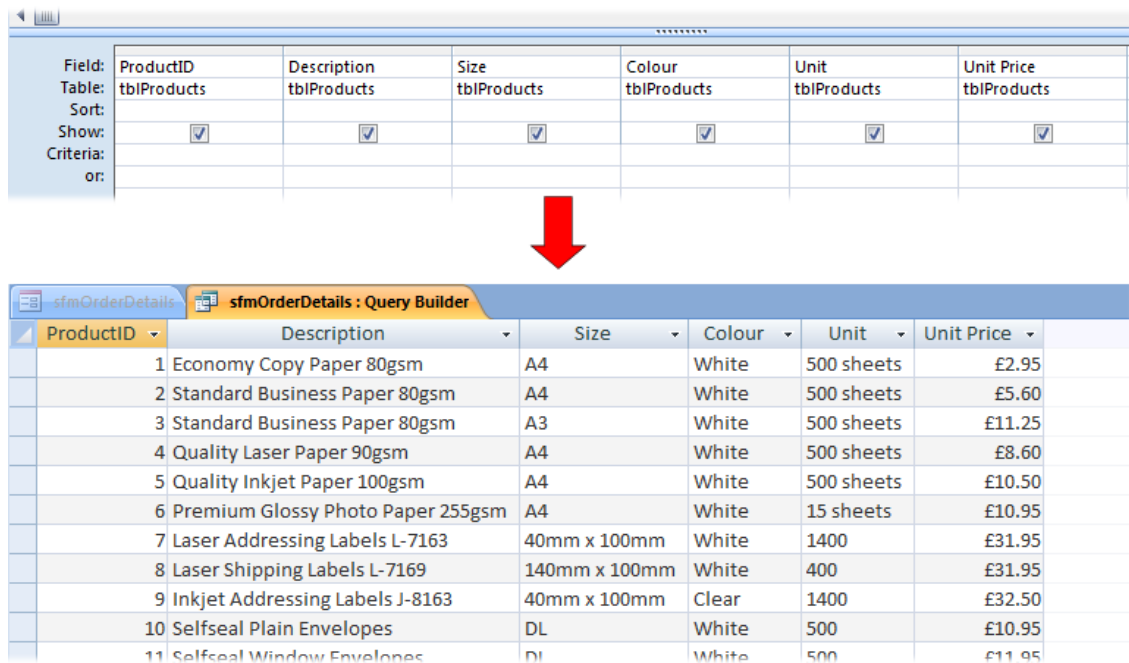


Fig. 11 The query creates 6 columns of data.

It is important to specify the various column widths and the total width of the list so that all the data is visible when the list is open (*Fig. 12*).

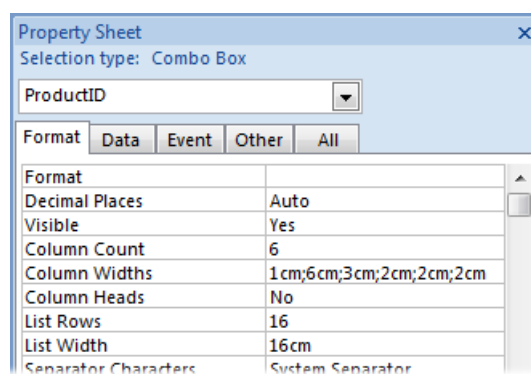


Fig. 12 The Column Count, Column Widths and List Width are specified in the Property Sheet.

The resulting combo box list shows all the information the user needs to choose the correct item (*Fig. 13*).

Note: With a multi-column list the value of the bound column is shown in the combo box after the user makes their choice (in this case the ProductID number). If you hide this column by setting its width to zero (0 cm) Access will display the value from first visible column instead. This is often more useful since the form will display some useful information whilst still entering the required number into the field.

ID	ProductID	Item	Quantity	Price
1	Economy Copy Paper 80gsm	A4	White	500 sheets £2.95
2	Standard Business Paper 80gsm	A4	White	500 sheets £5.60
3	Standard Business Paper 80gsm	A3	White	500 sheets £11.25
4	Quality Laser Paper 90gsm	A4	White	500 sheets £8.60
5	Quality Inkjet Paper 100gsm	A4	White	500 sheets £10.50
6	Premium Glossy Photo Paper 255gsm	A4	White	15 sheets £10.95
7	Laser Addressing Labels L-7163	40mm x 100mm	White	1400 £31.95
8	Laser Shipping Labels L-7169	140mm x 100mm	White	400 £31.95
9	Inkjet Addressing Labels J-8163	40mm x 100mm	Clear	1400 £32.50
10	Selfseal Plain Envelopes	D1	White	500 £10.95

Fig. 13 The combo box list shows all the required information.

With some creative query work you can combine data from several columns into one. In the next example the query selects just the *Sales Executives* names from the *tblStaff* table and combines their first and last names into a single field (Fig. 14). The list is then sorted by LastName then FirstName.

Field:	StaffID	SalesPerson: [LastName] & ", " & [FirstName]	LastName	FirstName	JobTitle
Table:	tblStaff		tblStaff	tblStaff	tblStaff
Sort:			Ascending	Ascending	
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Criteria:					"Sales Executive"
or:					

Fig. 14 This query creates a sorted list of salespeople.

The resulting **RowSource** data contains two columns, a column of StaffIDs and a column of names. By setting the width of the *StaffID* field to zero (Fig. 15) only the names are displayed in the combo box list and in the field after an item is chosen (Fig. 16) but the correct StaffID is still placed into the field.

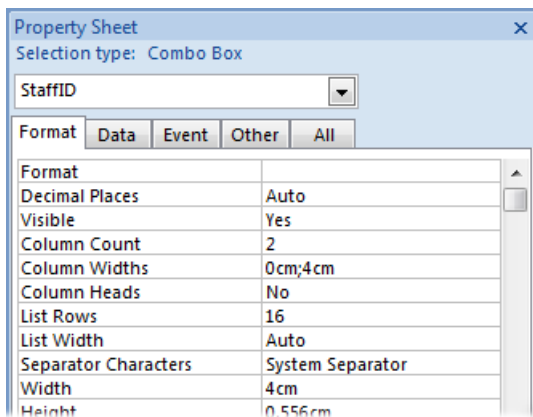


Fig. 15 The first column is hidden by setting its width to zero.

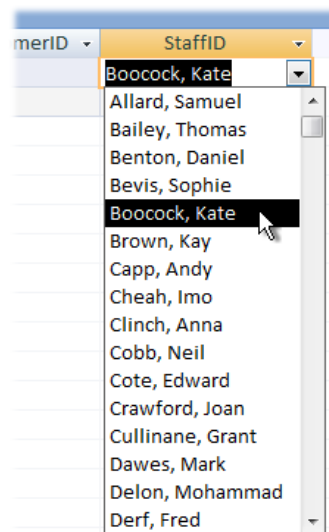


Fig. 16 Only the second column is displayed.