# Access Tutorial 8: Combo Box Controls
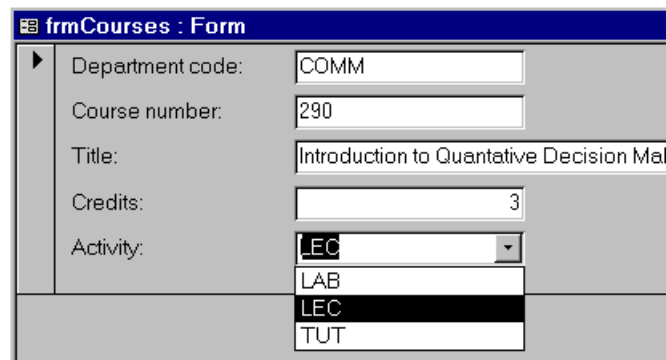
## 8.1 Introduction: What is a combo box?

So far, the only kind of "control" you have used on your forms has been the text box. However, Access provides other controls (such as combo boxes, list boxes, check boxes, radio buttons, etc.) that can be used to improve the attractiveness and functionality of your forms.

A combo box is list of values from which the user can select a single value. Not only does this save typing, it adds another means of enforcing referential integrity since the user can only pick values in the combo box. For example, a combo box for selecting course activities from a predefined list is shown in Figure 8.1.

Although advanced controls such as combo boxes and list boxes look and behave very differently than simple text boxes, their function is ultimately the

**FIGURE 8.1:** **A combo box for filling in the** `Activity` **field.**



same. For example, in Figure 8.1, the combo box is bound to the `Activity` field. When an item in the combo box is selected, the string (e.g., "LEC") is copied into the underlying field exactly as if you had typed the letters L-E-C into a text box.

⚠️ It is important to realize that combo boxes have no intrinsic search capability. Combo boxes change values—they do not automatically move to the record with the value you select. If you want to use a combo box for search, you have to program the procedure yourself (see Tutorial 15 for more details).

## 8.2   Learning objectives

❒ How do I create a bound combo box?

❒ Can I create a combo box that displays values from a different table?

❒ How do I show additional information in a combo box?

❒ How do I prevent certain information from showing in the combo box?

❒ Can I change the order in which the items appear in a combo box?

❒ What is tab order? How do I change it so that the cursor moves in the correct order?

❒ Should I put a combo box on a key field?

## 8.3   Tutorial exercises

• Open your `frmCourses` form in design mode.
• Ensure the toolbox and field list are visible (recall Figure 6.3).

### 8.3.1   Creating a bound combo box

Although Access has a wizard that simplifies the process of creating combo boxes, you will start by building a simple combo box (similar to that shown in Figure 8.1) with the wizard turned off. This will give you a better appreciation for what the wizard does and provide you with the skills to make refinements to wizard-created controls.

• Delete the existing `Activity` text box by selecting it and pressing the *Delete* key.

- The wizard toggle button (  ) in the toolbox allows you to turn wizard support on and off. Ensure the button is out (wizards are turned off).
- Click on the combo box tool (  ). The cursor turns into a small combo box.
- With the combo box tool selected, drag the `Activity` field from the field list to the desired location on the form's detail section, as shown in Figure 8.2.

The process of selecting a tool from the toolbox, and then using the tool to drag a field from the field list ensures that the control you create (text box, combo box, etc.) is bound to a field in the underlying table or query.

⚠️ If you forget to drag the field in from the field list, you will create an unbound combo box, as shown in Figure 8.3. If you accidently create

an unbound combo box, the easiest thing to do is to delete it and try again.

**FIGURE 8.3: An unbound combo box (not what you want).**
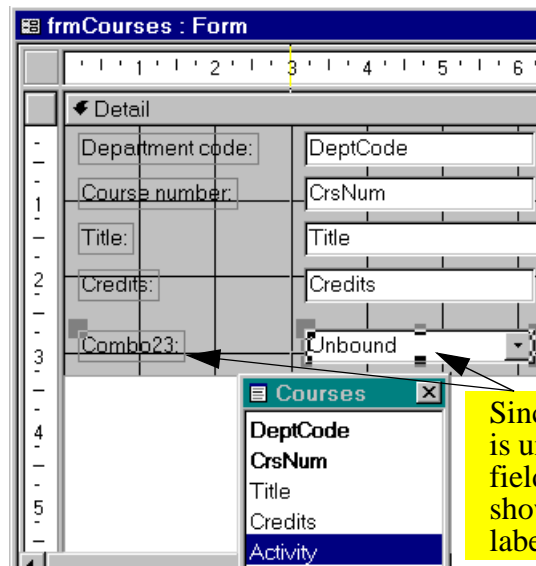


Since the control is unbound, no field name shows and the label is generic.

**FIGURE 8.2:** Create a bound combo box.
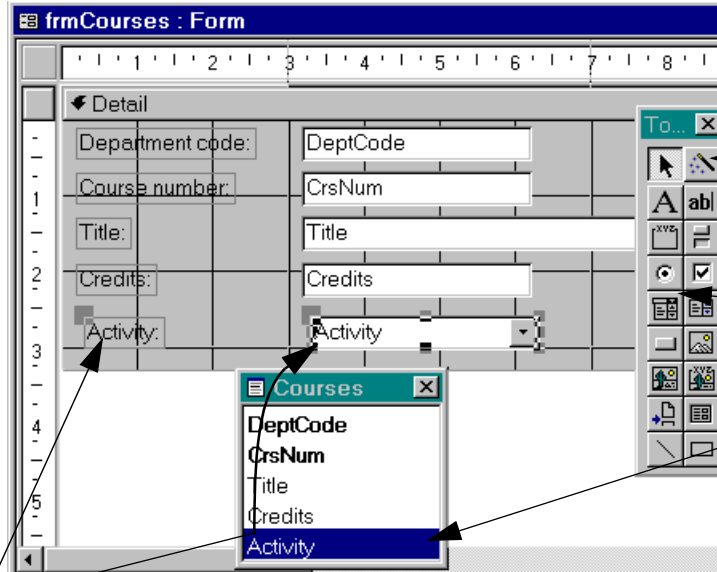


**a** Ensure the wizard button is not depressed.

**b** Click on the combo box button to activate the combo box tool.

**c** Select the `Activity` field from the field list.

**d** Drag the `Activity` field on to the detail area. If you have done this correctly, the name of the underlying field should show in the combo box and the label should take the value of the field's caption

## 8.3.2   Filling in the combo box properties

In this section, you will tell Access what you want to appear in the rows of new combo box.

• Switch to form view and test the combo box.

At this point, the combo box does not show any list items because we have not specified what the list items should be. There are three methods of specifying what shows up in the combo box list:

1. enter a list of values into the combo box's *Row Source* property;
2. tell Access to get the value from an existing table or query;
3. tell Access to use the names of fields in an existing table (you will not use this approach).

Although the second method is the most powerful and flexible, you will start with the first.

• Bring up the property sheet for the `Activity` combo box.

• Change the *Row Source Type* property to `Value List` as shown in Figure 8.4. This tells Access to expect a list of values in its *Row Source* property.

**FIGURE 8.4: Set the *Row Source Type* property.**
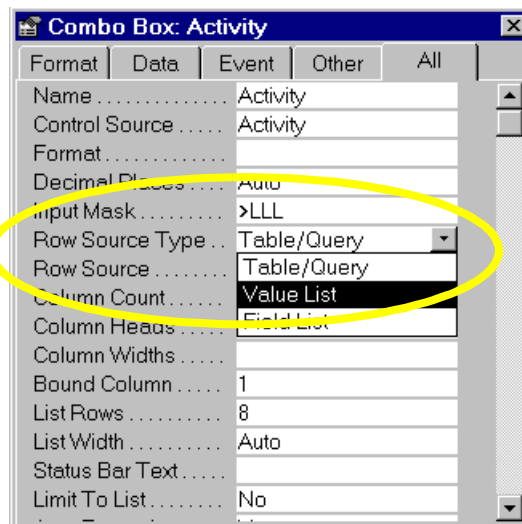
- Enter the following into the *Row Source* property:

  `LAB;LEC;TUT`

- Set the *Limit To List* property to `Yes`.

⚠️ If the *Limit To List* property is set to `No`, the user can ignore the choices in the combo box and simply type in a value (e.g., "SEM"). In this particular situation, you want to limit the user to the three choices given.

- Switch to form view and experiment with the combo box.

❓ Notice that the combo box has some useful built-in features. For example, if you choose to type values rather than select them with a mouse, the combo box anticipates your choice based on the letters you type. Thus, to select "TUT", you need only type "T".

### 8.3.3 A combo box based on another table or query

An obvious limitation of the value-list method of creating combo boxes is that it is impossible to change or update the items that appear in the list without knowing about the *Row Source* property.

A more elegant and flexible method of populating the rows of a combo box is to have Access look up the values from an existing table or query. Although the basic process of setting the combo box properties remains the same, it is more efficient to rely on the wizard when building this type of combo box.

Before you can continue, you need a table that contains appropriate values for course activities.

- Switch to the database window and create a new table called `Activities`.
- The table should consist of two fields: one called `Activity` and the other called `Descript`, as shown in .

**FIGURE 8.5: Create a table containing course activities.**



• Populate the table with the same values used in Section 8.3.2.

The result is a table containing all the possible course activities and a short description to explain the meaning of the three-letter codes. You can now return to creating a combo box based on these values.

• Delete the existing `Activity` combo box.
• Ensure the wizard button (⬛) in the toolbox is depressed (wizards are activated).
• Repeat the steps for creating a bound combo box (i.e., select the combo box tool and drag the `Activity` field from the field list on to the detail section). As shown in Figure 8.6, this activates the combo box wizard.

The wizard asks you to specify a number of things about the combo box:

1. the table (or query) from which the combo box values are going to be taken;

**FIGURE 8.6: Create a combo box using the combo box wizard.**



**a** Create a bound combo box.

**b** Have Access look up the values from a table or query.
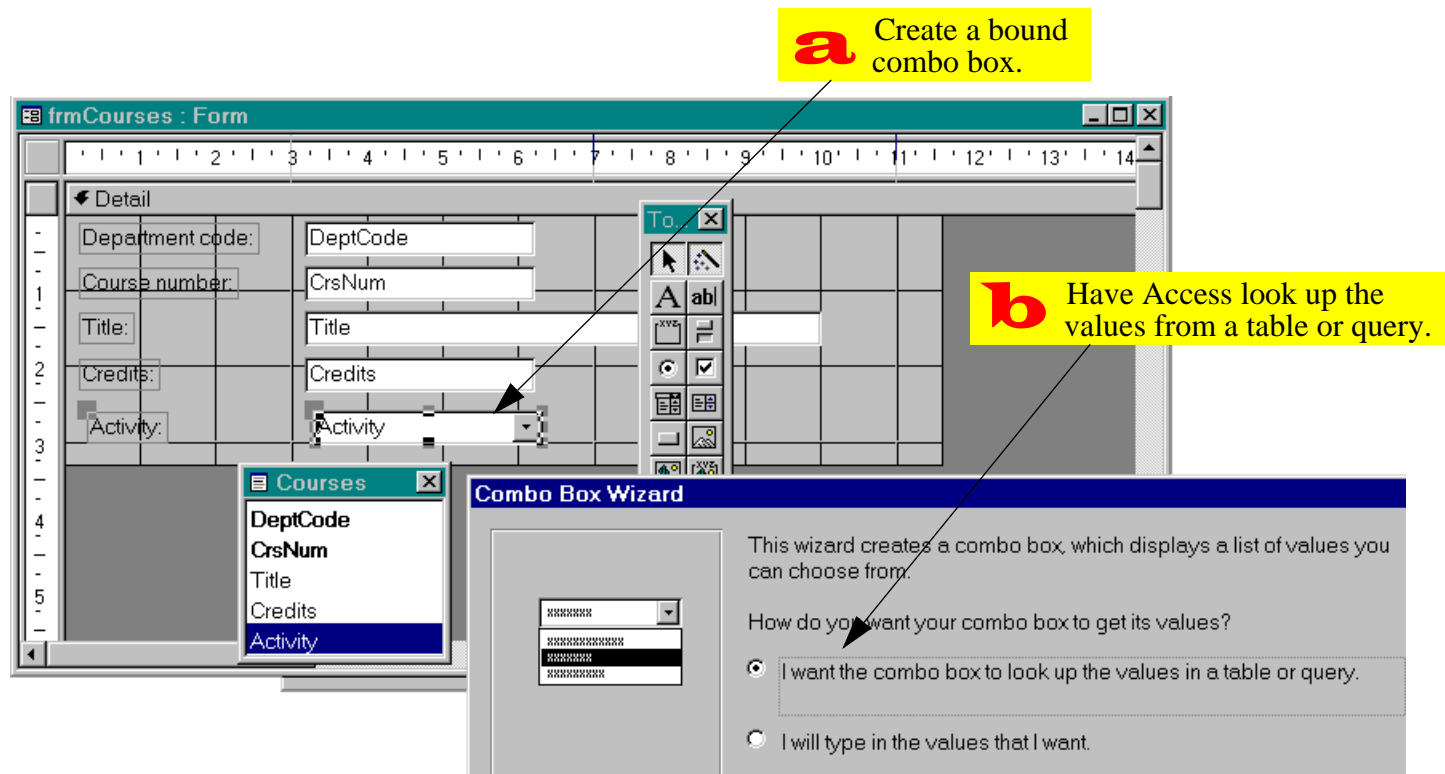
2.  the field (or fields) that you would like to show up as columns in the in the combo box;
3.  the width of the field(s) in the combo box (see Figure 8.7);
4.  the column from the combo box (if more than one field is showing) that is inserted into the underlying field; and,
5.  the label attached to the field (see Figure 8.8).

When you are done, the combo box should look similar to that shown in Figure 8.1. However, updating or changing the values in the combo box is much easier when the combo box is based on a table.

- Add "SEM" (Seminar) to the `Activities` table.
- Return to the form, click on the Activity combo box, and press *F9* to **requery** the combo box.
- Verify that "SEM" shows up in combo box.

⚠️ Access creates the rows in a combo box when the form is opened. If the values in the source table or query change while the form is open these changes are not automatically reflected in the combo box rows. As a consequence, you have to either (a) close and re-open the form, or (b) requery the form. Although you can automate the requery process, we will rely on the *F9* key for the time being.

### 8.3.3.1    Showing more than one field in the combo box

One problem the combo boxes created so far is that they are not of much use to a user who is not familiar with the abbreviations "TUT", "SEM", and so on. In this section, you will use the `Descript` field of the `Activities` table to make the combo box more readable, as shown in Figure 8.9.

- Delete the existing combo box and start again.

**FIGURE 8.7: Fill in the combo box
wizard dialog sheets.**

**Combo Box Wizard**

Which table or query should provide the values for your combo box?

Activities
Catalog View
Courses
Departments
Employees
Sections

**a** The new `Activities` tables contains the values for the combo box.

w:

Tables

Cance

**Combo Box Wizard**

Which fields contain the values you want included in your combo box?

The fields you select become columns in your combo box.

**b** The combo box can show more than one field. Select only `Activity` for now.

Available Fields:

Descript

Selected Fields:

Activity

>

**Combo Box Wizard**

How wide would you like the columns in your combo box?

To adjust the width of a column, drag its right edge to the width you want, or double-click the right edge of the column heading to get the best fit.

**c** Use the column selector (the grey bar at the top of the column) to resize the column to the desired width.

| Activity |
|----------|
| LAB |
| LEC |

**FIGURE 8.8: Fill in the combo box wizard dialog sheets (continued).**
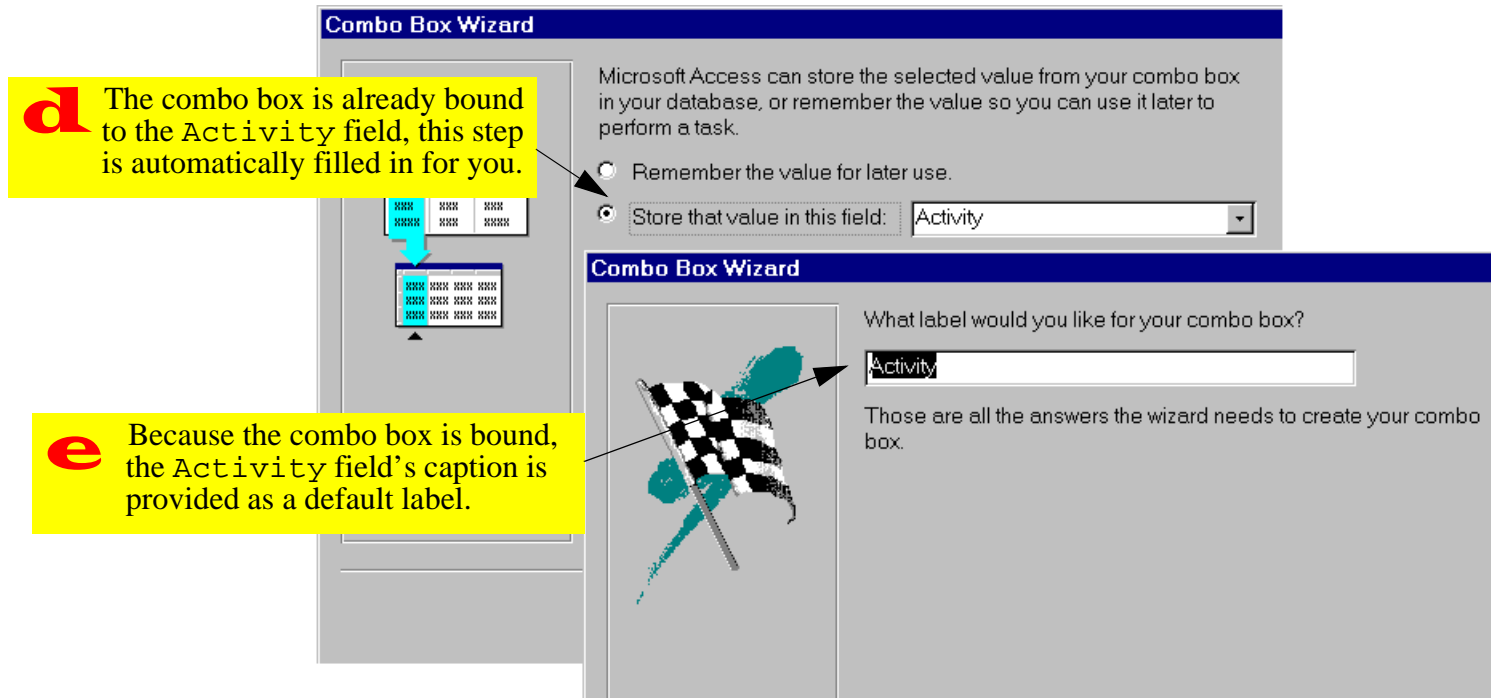


**d** The combo box is already bound to the `Activity` field, this step is automatically filled in for you.

**Combo Box Wizard**

Microsoft Access can store the selected value from your combo box in your database, or remember the value so you can use it later to perform a task.

○ Remember the value for later use.

◉ Store that value in this field:  | Activity |

**Combo Box Wizard**

What label would you like for your combo box?

| Activity |

Those are all the answers the wizard needs to create your combo box.

**e** Because the combo box is bound, the `Activity` field's caption is provided as a default label.

- Fill in the wizard dialog sheets as in Section 8.3.3 but make the changes shown in Figure 8.10.
- Verify that your combo box resembles Figure 8.9.

### 8.3.3.2    Hiding the key field

Assume for a moment that you, as a developer, do not want users to even see the three-letter abbrevia-tions and want them to select a course activity value based solely on the `Descript` field.

In such a case, you could include only the `Descript` column in the combo box. However, this would not work because the `Activity` field of the `Courses` table expects a three-letter abbreviation. As such, the combo box would generate an error when it tried to stuff a long description into the rela-tively short field to which it is bound.

In this section, you will create a combo box identical to that shown in Figure 8.9 except that the key col-umn (`Activity`) will be hidden from view. Despite its invisibility, however, the `Activity` column will still be bound to the `Activity` field of the underly-ing table and thus the combo box will work as it should.

- Delete the existing combo box and start again using the combo box wizard.

**FIGURE 8.10: Use the wizard to add more than one field to the combo box.**

## Combo Box Wizard

Which fields contain the values you want included in your combo box?

The fields you select become columns in your combo box.

Available Fields:

Selected Fields:

Activity
Descript

**a** Bring both fields from the `Activities` table into the combo box.

**b** Uncheck the "hide key" box and resize the columns appropriately. Note that Access version 2.0 does not have the "hide key" feature

## Combo Box Wizard

How wide would you like the columns in your combo box?

To adjust the width of a column, drag its right edge to the width you want, or double-click the right edge of the column heading to get the best fit.

**c** Select the column that provides the value of interest (in this case, `Activity`).

☐ Hide key column (recommended)

| | Activity | Descript |
|---|----------|----------|
| ▶ | LAB | Lab |
| | LEC | Lecture |
| | TUT | Tutorial |

## Combo Box Wizard

When you select a row in the combo box, you can store a value from that row in your database, or you can use the value later to perform an action. Choose a field that uniquely identifies the row.

Available Fields:

Activity
Descript

- Include both the `Activity` and `Descript` fields in the combo box.
- Resize the Activity column as shown in Figure 8.11. Note that users of version 7.0 can simply leave the "hide key" box checked—the result is the same.
- Ensure that the *Input Mask* property for the combo box (which is inherited from the field's *Input Mask* property) is blank.
- Verify that the resulting combo box resembles that shown in Figure 8.12.

⚠ Combo boxes with hidden keys can be confusing. The important thing to remember is that even though the description (e.g., "Lecture") now shows in the combo box, what is really stored in the underlying field is the hidden key (e.g., "LEC").

**FIGURE 8.12: A combo box with a hidden key.**



### 8.3.3.3 Changing the order of items in the combo box

A combo box based on a table shows the records in one of two ways:

1. If the table does not have a primary key, the records are shown in their natural order (that is, in the order they were added to the database).
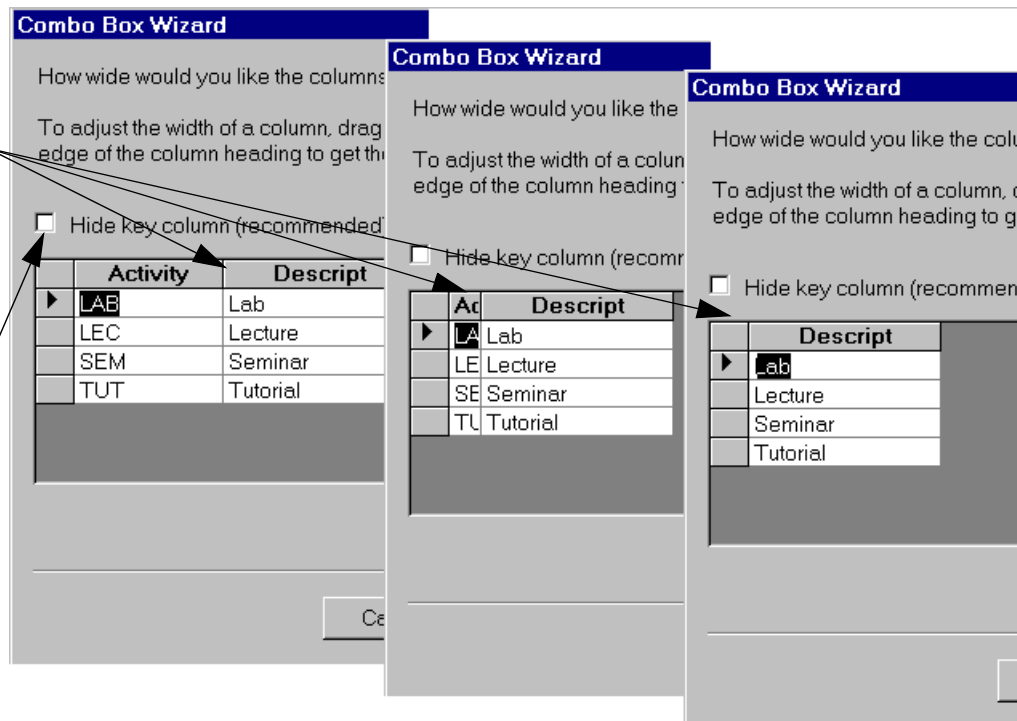
## FIGURE 8.11: Resize the columns to hide the key.

**a** Click on the right side of the column selector and drag the edge of the `Activity` column to the far left (i.e., make its width zero)

**(?)** Hiding the key is such a common operation that Access version 7.0 includes the "hide key" check box.

2.  If the table does have a primary key, then the records are sorted in ascending order according to the key.

It may be, however, that you want a different order within the rows of the combo box. To achieve this, you can do one of two thing:

1.  Create a stand-alone query (in which the sort order is specified) and use this query as the source for the combo box.
2.  Modify the "ad hoc" query within the *Row Source* property of the combo box.

If you intend to make several major changes to the basic information in the underlying table (e.g., joins, calculated fields), it is usually better to create a stand-alone query. In this way, the same query can be used by many combo boxes.

If the changes are quite minor (for instance, sorting the records in a different order), you may prefer to modify the *Row Source* property.

In Section 8.3.2, you set the *Row Source* property to equal a list of values. When the combo box is based on values from a table or a query, however, the *Row Source* is an SQL statement (recall Tutorial 5) rather than a list of values. You can either edit the SQL statement directly or invoke the QBE editor.

In this section, you will order the items in you combo box according to the length of the `Descript` field (this is done merely for illustrative purposes).

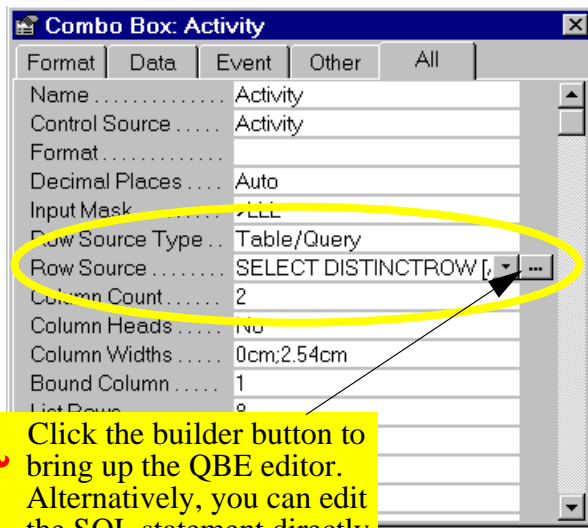- Bring up the property sheet for the `Activity` combo box.
- Put the cursor in the *Row Source* property. As shown in Figure 8.13, a builder button ( ▦ ) appears.
- Press the builder button to enter the "SQL builder" (i.e., the QBE editor).

### FIGURE 8.13: Invoke the builder for the *Row Source* property.



**a** Click the builder button to bring up the QBE editor. Alternatively, you can edit the SQL statement directly.

- Create a calculated field called `DescLength` using the following expression:

```
DescLength: Len([Descript])
```

(`Len()` is a built-in function that returns the length of a string of characters).

- Sort on `DescLength` in descending order.
- Switch to datasheet view to ensure the query is working as it should.
- Ensure the *Show* box for the field is unchecked, as shown in Figure 8.14.
- Instead of saving the query in the normal way, simply close the QBE box using the close button (⊠).

⚠ If you save the query, it will be added to your collection of saved queries (the ones that are displayed in the database window). However, if you simply close the QBE window, the *Row Source* property will be updated and no new database object will be created.

**FIGURE 8.14: Use the QBE editor to modify the *Row Source* property.**
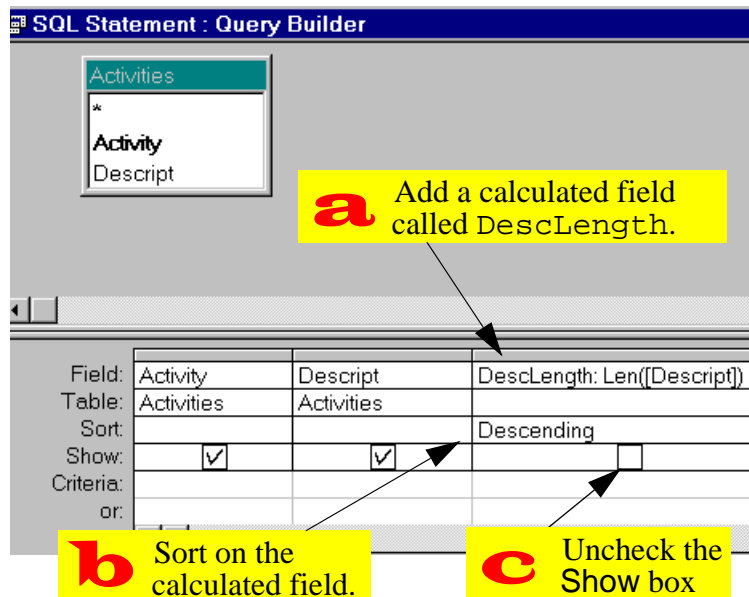


### 8.3.4   Changing a form's tab order

A form's **tab order** determines the order in which the objects on a form are visited when the *Tab* or *Enter* (or *Return*) keys are pressed. Access sets the tab order based on the order in which objects are added to the form. As a result, when you delete a text box and replace it with a combo box or some other control, the new control becomes the last item in the tab order regardless of its position on the form.

To illustrate the problem, you are going to create a combo box for the DeptCode field.

- Delete the DeptCode text box and replace it with a combo box based on the Departments table.
- Switch to form view. Notice that the focus starts off in the CrsNum field instead of the DeptCode field.
- Press tab to move from field to field. Notice that after DeptCode is left, the focus returns to the CrsNum field of the next record.

- To fix the problem, return to form design mode and select *View > Tab Order* from the main menu.

⚠️ ② In Access version 2.0, the menu structure is slightly different. As such, you must select *Edit > Tab Order*.

- Perform the steps in Figure 8.15 to move `Dept-Code` to the top of the tab order.

## 8.4   Discussion

### 8.4.1   Why you should never use a combo box for a non-concatenated key.

A mistake often made once new users learn how to make combo boxes is to put a combo box on every-thing. There are certain situations, however, in which the use of a combo box is simply incorrect.

For example, it never makes sense to put a combo box on a non-concatenated primary key. To illustrate this, consider the `Departments` form shown in Figure 8.16. On this form, the `DeptCode` text box has been replaced with a combo box that draws its values from the `Departments` table.

**FIGURE 8.16: A combo box bound to a key field.**



This combo box appears to work. However, if you think about it, it makes no sense: The form in Figure 8.16 is a window on the `Departments` table. As such, when the `DeptCode` combo box is used,

**FIGURE 8.15: Adjust the tab order of fields on a form.**

**b** Drag the record selector to the desired position in the list.

**a** Click on the record selector of the field you wish to move.

**Tab Order**

Section
- ○ Form Header
- ● Detail
- ○ Form Footer

Custom Order:
- CrsNum
- Title
- Credits
- Activity
- DeptCode

Click to select a row, or click and drag to select multiple rows. Drag selected row(s) to move them to desired tab order.

OK    Cancel

**Tab Order**

Section
- ○ Form Header
- ● Detail
- ○ Form Footer

Custom Order:
- DeptCode
- CrsNum
- Title
- Credits
- Activity

Click to select a row, or click and drag to select multiple rows. Drag selected row(s) to move them to desired tab order.

OK    Cancel    Auto Order

**?** For forms in which the fields are arranged in a single column from top to bottom (such as this one), you can press Auto Order to order them automatically.

one of two things can occur depending on whether a new record is being created or an existing record is being edited:

1. **A new record is being created** — If a new record is being created (i.e., a new department is being added to the information system), a unique value of `DeptCode` must be created to distinguish the new department from the existing departments. However, the combo box only shows `DeptCode` values of *existing* departments. If the *Limit To List* property is set to `Yes`, then the combo box prevents the user from entering a valid `DeptCode` value.

2. **An existing record is being edited** — It is important to remember that a combo box has no intrinsic search capability. As such, selecting "CPSC" in the `DeptCode` combo box does not result in a jump to the record with "CPSC" as its key value. Rather, selecting "CPSC" from the combo box is identical to typing "CPSC" over whatever is currently in the `DeptCode` field. This causes all sorts of problems; the most obvious of these is that by overwriting an existing value of `DeptCode`, a "duplicate value in index, primary key, or relationship" error is generated (there is already a department with "CPSC" as its `DeptCode`).

Note that a combo box may make sense when the key is concatenated. An example of this is the `DeptCode` combo box you created in .

## 8.4.2 Controls and widgets

Predefined controls are becoming increasingly popular in software development. Although Microsoft includes several predefined controls with Access (such as combo boxes, check boxes, radio buttons, etc.), a large number of more compex or specialized controls are available from Microsoft and other ven-

dors. In addition, you can write your own custom controls using a language like Visual C++ or Visual Basic and use them in many different forms and applications.

An example of a more complex control is the calendar control shown in Figure 8.17. A calendar control can be added to a form to make the entry of dates easier for the user. Microsoft calls such components "ActiveX controls" (formerly known as "OLE controls"). Non-microsoft vendors provide similar components but use different names, such as "widgets".

There are two main advantages of using controls. First, they cut down on the time it takes to develop an application since the controls are predefined and pre-tested. Second, they are standardized so that users encounter the same basic behavior in all applications.

## 8.5   Application to the assignment

There are a number of forms in your assignment that can be greatly enhanced by combo boxes.

- Create a combo box on your order form to allow the user to select customers by name rather than `CustID`. Since your `CustID` value is a counter, it has no significance beyond its use as a primary key. Generally, such keys should be hidden from view.
- Create a combo box in your order details subform to allow the user to select products. Since the `ProductID` values are used by both you and your customers, they have some significance beyond the information system. As such, `ProductID` should be visible in all combo boxes. In addition, the items in the product list should be sorted by `ProductID`. This makes it easier to select a product by typing the first few numbers.
- Create combo boxes on other forms as required.

**FIGURE 8.17: A calendar control on a form.**



The calendar control can be bound to date/time fields, thereby making it easier for users to enter dates.

Like other objects in Access, controls have properties and events that determine the appearance and behavior of the control.