# LO10 – Apply Advanced Functionality to Forms and Reports

## LO10.1 Populate Form Controls Based on Dynamic Queries

## Why Use Dynamic Queries?

We are using Access, but regardless of the tools we are using (we could be using the php programming language and a MySQL database) the most powerful thing we can do is to create dynamic queries. Dynamic queries added to our forms and reports allow users to dynamically create a result set for themselves. It would sure suck if the CEO wanted a report on the last 50 orders but the system could only print out all the orders since the inception of the company.

To accomplish this, we will allow the user to pick from available options and, based on those options, display the corresponding result set. Thus, the user is going to provide **input** that will feed our queries to limit/filter the results.

Dynamic queries can be used to create forms and reports that populate data based on input from a user.

### Create a form with two combo boxes (KelseyVets.accdb)

We need to create a form with the two combo boxes with the following:

* We need a combo box that lists all the **Provinces** and one that lists all **Customers**
* The values selected from the combo boxes will be used by the query(s) to populate other combo boxes and list boxes
* We will grab provinces from the **Provinces** table.
* Data source for the customer combo box is the Customer table **BUT** it will be dynamic (loads only customers that are from a selected province)

**Follow these steps:**

1. Open the **KelseyVets.accdb** database
2. Open a new form with the following properties:
   1. Size of 5.5” by 4”
   2. Caption: “*Create a dynamic query by selecting criteria from combo boxes*”
   3. Record Selectors, Navigation Buttons, and Scrollbar to No or Neither
   4. Save as **frmPets**
3. Create a **Province** combo box using the *wizard*.
   1. Turn on the wizard
   2. Place a combo box in the upper left corner of the form.
   3. Select “**I want the combo box to Look Up the values in a table or query**”
   4. Select Province table
   5. Select **ProvinceName** as the only field (note: it automatically takes the PK)
   6. Sort ascending by **ProvinceName**
   7. Make sure “**Hide Key Column**” is checked off
   8. Adjust the width so that the longest Province can be fully viewed
   9. Caption property of the label: **Province**
   10. Name property of the combo box: **cboProvince**
   11. Name property of the label: **lblProvince**.
4. Create a **Customer** combo box not using the wizard
   1. Turn off the wizard
   2. Place a combo box in the upper right corner of the form
   3. Set the following properties:
      1. Name: **cboCustomer**
      2. Row Source: build a query that returns **Customer Number, Customer Name**
      3. Column Count: 2
      4. Column Widths: 0”;1”
      5. Limit to List (on data tab): Yes (This property will limit a combo box’s values to the listed items).
   4. Set the following properties of the label:
      1. Name: **lblCustomer**
      2. Caption: “Customer”
5. Let’s view the form so far
   1. A picture containing diagram

      Description automatically generated
6. Change for **Form View**. Look carefully at the values in the two lists. Note that there is a problem: the **Customers** names are all displayed regardless of whether they are from the selected province or not. This is because our query is bringing back everything.
7. Let’s make it so that when the province name changes, the options displayed in the customer box change as well. What event should we use? “After Update” on the province combo box.
8. Go back and modify the query to obtain the province and set criteria to ‘AB’:
   1. “SELECT tblCustomer.[Customer Number], tblCustomer.[Customer Name] FROM tblCustomer WHERE (((tblCustomer.Province)="AB"));”
9. Copy the Row Source property from the Customer Combo Box (you can use shift-end)
10. Let’s hook into the Customer combo box After Update event. This will generate a new subroutine for that event.

|  |
| --- |
| Option Compare Database  Option Explicit  Private Sub cboProvince\_AfterUpdate()  'Copy and paste the SQL statement into the code. We want to display  ' only customers who are from the province that has been selected.  ' We will need to set the province = to the value of the province combo box    'SELECT tblCustomer.[Customer Number], tblCustomer.[Customer Name] FROM tblCustomer  ' WHERE tblCustomer.Province='AB';    Dim strFinalSQL As String    Const sqlCodeBegin = "SELECT tblCustomer.[Customer Number], tblCustomer.[Customer Name] FROM tblCustomer WHERE tblCustomer.Province='"  Const sqlCodeEnd = "'"    'Concatenate the actual selected province with our SQL statement from above  strFinalSQL = sqlCodeBegin & Me.cboProvince.Value & sqlCodeEnd    'Set the rowsource of the customer combo box to the string we just generated  Me.cboCustomer.RowSource = strFinalSQL  End Sub |

1. Try it. Notice how the customer name stays selected as we change the different provinces. We probably don’t want that to happen. Also – it might be easier to look through the customer names if they were in order. Let’s sort them as well.

**Error Checking**

Creating Dynamic SQL is Hard, and requires a lot of patience and familiarity with debugging tools.

Create breakpoints in your subroutine so that you can inspect the SQL that is being generated. To view the values our variables, you can use Immediate Window with the following syntax:

?<variableName>

For example, to view our final sql code, I would type in **?strFinalSQL**

## Add a List Box to the form

We need to create a list box to display the Pets

* In order to design the list box, we will create a snapshot of what the list box should look like.
* We will add a static query to the list box and then remove it later when we have the VBA code in place to make the query dynamic based on the selections in the combo boxes.

In the list box that we add, we would like to see **PetName, PetID, Breed, Animal Type** for all pets that match the combo box selections.

1. Consider whether we can get all the fields we want from any one table? The answer is no. We get the **PetID, PetName,CustomerNumber** from **tblPet**. We get the **AnimalType** from **tblAnimalType** table. We need to create a query to select the list box values.
2. Create a new query as shown here:
   1. Graphical user interface, application

      Description automatically generated
   2. Save as **qryPetValues**
   3. Close the query and switch back to the form **frmPet**
3. Add a list box, again:
   1. Turn on the wizard
   2. Place a list box below the two combo boxes
   3. Select “I want the combo box …”
   4. Select **qryPetValues**
   5. Select **Customer Number**, **Pet ID, Pet Name, AnimalTypeID, Type of Animal, Breed**. (Customer Number is a dynamic value so we need it). We need Pet ID because we will need it later to make another dynamic listbox and we need AnimalTypeID because it’s a FK to another table.
   6. Sort by Pet Name
   7. Make sure all columns are wide enough to display the longest value in the column
   8. Select PetID as the field that will uniquely identify a row.
   9. Caption: “Pets by Customer and Province”
   10. Set the following properties of the list box:
       1. Name: lstPet
       2. Column Heads: Yes
   11. Set the following properties of the list box label:
       1. Name: lblPet
   12. View the form and make any adjustments
   13. In Design View:
       1. A screenshot of a computer

          Description automatically generated with medium confidence
   14. In Form View:
       1. Table

          Description automatically generated
4. Double check all your combo boxes and the list box to see if you need to set the first column widths to 0”
5. Can resize some columns if needed.
6. Save the form.

# 10.2 Create Dynamic Queries Based on Form Selections

## Add Dynamic Queries to The Form

We need some method of connecting the values selected in the combo boxes to the records that are displayed in the list box. We can do this by having a WHERE clause in the Row Source property of **lstPets** list box set up to be equal to the value selected in the **Customer** combo box:

Take a look at **qryPetValues**. Again, we want a WHERE clause so that we can specify that we only want to see pets from certain customers.

Follow the steps:

1. Copy the (adjusted, with a WHERE clause for customer number) SQL from the query builder of **qryPetValues**
2. First we will split the SQL statement apart and assign the parts that DON’T change to constants. **Remember to include the single quotation marks in the constant.**
3. Open the VBA Editor
4. Create a new public procedure, FillPetList, in the form class module
5. Have the following code:

|  |
| --- |
| Public Sub FillPetList()  'This procedure will create a SQL statement that will populate the list box, lstPet  ' This will be assigned to the RowSource property  ' It will also refresh the list box by applying a Requery method and  ' change the caption of the list box label to reflect the WHERE clause criteria    ' INNER JOIN tblPets ON tblAnimalType.AnimalTypeID = tblPets.AnimalTypeID WHERE tblPets.[Customer Number]="AC001";  Dim strFinalSQL, strPetCaption As String    Const strSQLSelect = "SELECT tblPets.[Customer Number], tblPets.[Pet ID], tblPets.[Pet Name]," & \_  " tblAnimalType.AnimalTypeID, tblAnimalType.[Type of Animal], tblPets.Breed FROM tblAnimalType" & \_  " INNER JOIN tblPets ON tblAnimalType.AnimalTypeID = tblPets.AnimalTypeID " & \_  " WHERE tblPets.[Customer Number] = '"    Const strSQLOrder = "' ORDER BY [Pet Name]"    strFinalSQL = strSQLSelect & Me.cboCustomer.Value & strSQLOrder    Me.lstPet.RowSource = strFinalSQL  Me.lstPet.Requery    End Sub |

1. Notes:
   1. “& \_” at the end of the three lines is used to concatenate the rest of the string (&) and the space underscore ( \_) says it is continued on the next line.
   2. ‘” at the of strSQLSelect and the beginning of strSQLOrder is actuall a single quote (‘) followed by a double quote (“). Remember, the single quote must be included in the constant because all string values must be enclosed in single quotes.
2. Save the code.

## Add Events to the Form

Every time the user selects either a new customer or new province, we would like the list box to change. In order to do this we will call the FillPetList procedure in the AfterUpdate events of both combo boxes.

Follow these steps:

1. Create a new **AfterUpdate** event for **cboCustomer**
2. Enter the following code:
   1. **Call FillPetList**
3. Save the code
4. Run the form and try selecting some values.

## Tips for Populating combo boxes dynamically

1. If fields are required from ONLY one table, you can use the combobox wizard to bind the combobox to that table. (this will automatically create a query and put the SQL in the Rowsource)
2. If fields are required from more than one table, create a query using the query designer. The query should return the fields that you will want to display as well as include a sample criteria (ie: if you want to only return customers who live in a specified province, set your query to return all rows where province is set to Alberta).
3. Bind the query to the combo box. Note: this will be done for you automatically if you use the wizard to create the combobox.
4. When the query is bound to the combobox, the SQL for the query appears in the RowSource. Copy this SQL to use in your VBA code.
5. Criteria is always set in the “Where” clause of the SQL statement. This is where you will set a field equal to a value obtained from another control (usually a combobox or listbox).
6. In SQL you must always put string values inside single quotes. Example: ‘Alberta’
7. In SQL numeric values do **NOT** go between single quotes.
8. If you create an alias in your query, you will use the alias name in your SQL. If you do not create an alias, you will use the actual column name.
9. Only field names in the query that have spaces in the name must be enclosed in square brackets. Example: [Customer Name].

## Tips for writing VBA code

1. While it will work to create your SQL as one long string, it will be easier to troubleshoot if you break the string up into 3 sections. The first and the last sections are constants to decrease the chance of accidentally changing the values in the code later on:
   1. Start of the SQL code (Including SELECT, FROM, JOINS, start of WHERE clause)
   2. Criteria value(s) (This is where we would use Me.MyControl.Value to inject the actual user-selected value)
   3. End of the SQL code (closing single quote and / or ORDER BY clause)

\*Note: if you have more than one criteria to check, you could break the string up into more sections.

1. Use: & \_ at the end of a line to concatenate it with the following line. NOTE: there is a space between the text and the &. There is a space between the & and the \_.
2. When creating your SQL string text, do not forget that any criteria that is a string must appear between single quotes. This means the starting string will end with a single quote and a double quote and the end string will start with a double quote and an end quote. Example:
   1. startString = “Select \* from Table Where field=’”
   2. endString = “’ Order by field”

## Change the list box caption

We will dynamically change the caption of the list box to reflect the selections made in the combo boxes or if not items were matched.

To create the dynamic caption, we will introduce two new properties:

## Useful control properties

1. **The Column** property of a list or combo box: This property will return the value of the specified column. The Column property is zero based. We use this property when we want to display a column that is not the bound column.

**Example: Control.Column(column, row)**  You can use the **Column** property to refer to a specific column, or column and row combination, in a multiple-column combo box or list box. Use 0 to refer to the first column, 1 to refer to the second column, and so on. Use 0 to refer to the first row, 1 to refer to the second row, and so on.

1. **The ListCount** property of a list or combo box: This property will return the number of items in the list. We will use this property to determine which message to display.

**Example: Control.ListCount**  You can use the **ListCount** property to determine the number of rows in a list box or the list box portion of a combo box.

1. **Selected** property sets or returns whether a row is selected

**Example: Control.Selected(itemNumber)**  You can use the **Selected** property to select items in a list box. OR You can use the Selected property in VBA to determine if an item in a list box is selected. (-1 means it is selected and 0 means it is not)

1. **ItemData(index)** property enables you to iterate through the list of entries in a combo or list box. It returns the data in the bound column for the specified row (the index).

Follow these steps:

1. After the Requery statement, add the code below.

|  |
| --- |
| If Me.lstPet.ListCount = 0 Then  strPetCaption = "No Pets match the selected criteria"  Else  strPetCaption = "Pets from " & Me.cboProvince.Value & " Belonging to " & Me.cboCustomer.Column(1)  End If    Me.lblPet.Caption = strPetCaption |

1. Save the code
2. Run the form and test. (Look in Saskatchewan for Tanya Lung who has no pets)
3. Notice that it shows the short form for province. Go back and change from Me.cboProvince.Value to Me.cboProvince.Column(1)

## Add the Form Load Procedure

When the form opens, we want to empty all the controls.

1. Create a new **Form\_Load** event for **frmPets.**
   1. This event is available through the Events tab when we select Form properties.
2. Use the following code:

|  |
| --- |
| Private Sub Form\_Load()  'When the form opens all the controls will be cleared and  ' a default caption will be set for the list box    Me.cboProvince.Value = Null  Me.cboCustomer.Value = Null    Me.cboCustomer.RowSource = ""  Me.cboCustomer.Requery    Me.lstPet.RowSource = ""  Me.lstPet.Requery    Me.lblPet.Caption = "Please select a Province from the list"  Me.cboProvince.SetFocus    End Sub |

1. Save the code, run the form and test.

## Make the form more user-friendly

We will change the captions of the list box to provide the user with some instructions as to how to the use the form:

* When the form opens the caption will instruct the user to select a **Province**. This is already done.
* If the user selects a **Province**, but has not selected a Customer, the caption will instruct the user to select a **Customer**
* If the user has selected both a **Province** and **Customer**, then the caption will display the information regarding the selections (mostly done)

We will also not call **FillPetList** procedure unless both a province and customer have been selected.

1. Edit the **AfterUpdate** event for **cboCustomer**
2. Add the code shown below
3. Note that this message is exactly the same as the message assigned to the lblPet.Caption in the **Form\_Load** event procedure. To make our code more updateable we will create a **Global Constant** containing this string.
4. At the top of **frmPets** form class module, add the following global constants:

|  |
| --- |
| Option Compare Database  Option Explicit  Private Const strSelectCustomerMsg = "Please select a Customer from the list"  Private Const strSelectProvinceMsg = "Please select a Province from the list" |

1. Edit the **Form\_Load** event to use this new global constant:
   1. Me.lblPet.Caption = strSelectProvinceMsg
2. Edit the **AfterUpdate** event for **cboProvince**. Add the code shown below:
   1. Me.lblPet.Caption = strSelectCustomerMsg
3. Save the code, run the form, and try selecting some values.

**Complete Exercise 10.2.1**

## Possible SQL Errors in VBA Code

1. Check the code for syntax errors.
2. With the form in **Design View** delete the list box's**lstOrder**and**Record Source**, and then run the form again. Sometimes an incorrectly formed Record Source will get stuck and the VBA code will not be able to replace it. So even if your code is correct, the list box will not be able to display it.
3. Check the **Row Source**property for syntax errors:
4. Make sure all SQL Statement reserved words have spaces around them. Reserved words: SELECT, FROM, WHERE, ORDER BY, AND, etc.   
     
   **INCORRECT (errors in RED)**
5. SELECT CompanyName, OrderID, OrderDate, ShippedDat**eF**ROM qryCountryAndProduct

WHERE Country = 'Canada' AND ProductID = 17 ORDER B**YC**ompanyName

**CORRECT**

SELECT CompanyName, OrderID, OrderDate, ShippedDate FROM qryCountryAndProduct

WHERE Country = 'Canada' AND ProductID = 17 ORDER BY CompanyName

1. The semi-colon at the end of the SQL Statement is not required if it is a single SELECT Statement.

# 10.3 Call VBA Modules Based on form Events

## About VBA Modules and Form Events

Over the next few pages, we will alter the **frmPetVisits** form to allow it to display not only the visits but also the visit details. If we select of the visits from the existing list box, a second list will display all the visit details, related to the selected visit.

## Create a list box to display the visit details.

**Step 1: Create the stored query**,.

This query will be used to dynamically populate the Pet Visits list box. We need to include the columns that we wish to see in that list box (**Treatment, Treatment Price, Medication Price, Visit Type)** as well as the bound column from the **lstVisit** list box (**Visit Number)**.

* Create a new query as shown below:
* Graphical user interface, application

  Description automatically generated
* The **Total** column is an expression formatted to currency:
  + Total Cost: FormatCurrency([tblVisitDetails]Treatment Price]+[tblVisitDetails]![Medication Price])
* Save as **qryLineItems.**
* Close the query.

**Step 2 – Create the list box**

1. Open the **frmPetVisits** form in Design View
2. Add a list box below the **lstPet** list box
   1. Expand the form to make room.
   2. Turn on the Wizard
   3. Place a list box below the **lstPet** list box
   4. Select **I want the combo box to Look Up the values in a table or query**
   5. Select query: **qryLineItems**
   6. Select **All Fields** (**Visit Number** is in the query because it will become the dynamic value that changes based on the list box selection
   7. Sort by **Visit Type**
   8. Make sure all columns are wide enough to display the longest value in the column.
   9. Select **Visit Number** as the field that uniquely identifies the row.
      1. Caption: “Visit Details”
   10. Set the following properties of the list box:
       1. Name: **lstLineItem**
       2. Column Heads: Yes
   11. Set the following properties of the list box label:
       1. Name: lblLineItem
   12. View the form and make any adjustments necessary.
       1. **![Table

          Description automatically generated with medium confidence**
3. Save the form

## Create Dynamic Queries Based on Form Selections

When the user double clicks on the visits in the **lstVisits** list box, all the corresponding Visit Details (or line items) will be displayed in the **lstLineItem** list box.

To accomplish this, we assign the value in the **Row Source** property WHERE clause to be equal to the value clicked on in the **lstVisits** list box. In the stored query, one of our columns have aliases **(Treatment Price as Treatment Cost**). If there exists an alias you can use the alias, otherwise, you use the actual column name as created in the table. Take a look at the SQL for **qryLineItems**. Notice the **AS** keyword to define an alias.

NOTE: Only names containing spaces must be enclosed in square brackets: [ ].

Example:

From the list of all the visits, we will select the first visit to display. Select **Pet** is Bobo (AC001-01) and the Visit Number is 19980804-01. Follow these steps:

1. First we split the SQL statement apart and assign the parts that DON’T change to constants. Grab the SQL from qryLineItems and paste in VBA. We’ll create a new public procedure.
2. Open up the class module for our form, goto Insert, and then Procedure. Use the following code.

|  |
| --- |
| Public Sub FillLineItemList()  Const strSQLSelect = "SELECT \* FROM qryLineItems WHERE [Visit Number] = '"  Const strSQLOrder = "' ORDER BY [Visit Type]"    Dim strFinalSQL As String    strFinalSQL = strSQLSelect & Me.lstVisits.Value & strSQLOrder    Me.lstLineItem.RowSource = strFinalSQL  Me.lstLineItem.Requery    Me.lblLineItem.Caption = "Line items for Visit Number " & Me.lstVisits.Value    End Sub |

I find this a little easier to work with instead of multi-line dynamic SQL contained in strings. A good tip: Try to make a saved query and just select items from that query.

## Call the FillLineItemList Procedure

Add the events that will call the procedure written above. When the user double-clicks on a visit, all the corresponding visit details will be displayed in the **lstLineItem** list box

STOPPED HERE 04/13/2022

1. Create a new **DBLClick** event for **lstVisit:**
   1. From the Object List, select **lstVisit**
   2. The default **BeforeUpdate** event is added to the VBA code. We do not want this event.
   3. From the procedure list, select **DBLClick**. We want the double-click event.
2. Enter the following code:

|  |
| --- |
| Private Sub lstVisits\_DblClick(Cancel As Integer)  Call FillLineItemList  End Sub |

1. Save the code
2. Run the form and test.

## Clear the ListBoxes when the combobox values change

We need to clear the line item list box when setting new query criteria and change the label caption. We need to do the exact same clear processes in four places:

* cboProvince\_AfterUpdate event
* cboCustomer\_AfterUpdate event
* cboPetName\_AfterUpdate event
* OnLoad event

But instead of duplicating code we will create a single procedure that can be called by any of these events.

1. Create a new procedure **ClearListBox** and enter the code below:

|  |
| --- |
| Private Sub ClearListBox(lstBoxToClear As ListBox, lblOnListBox As Label, strCaption As String)  ' This procedure will clear the contents of the list box passed in and set  ' the list box's caption property to the value of strCaption    lstBoxToClear.RowSource = ""  lstBoxToClear.Requery    lblOnListBox.Caption = strCaption    End Sub |

1. Save the code
2. Create a fourth Global Constant at the beginning of our class module:
   1. **Private Const strLineItemMsg = "Visit Details"**
3. Add the clear list box call to the **cboProvince\_AfterUpdate code**:

|  |
| --- |
| Private Sub cboProvince\_AfterUpdate()  'Copy and paste the SQL statement into the code. We want to display  ' only customers who are from the province that has been selected.  ' We will need to set the province = to the value of the province combo box    'SELECT tblCustomer.[Customer Number], tblCustomer.[Customer Name] FROM tblCustomer  ' WHERE tblCustomer.Province='AB' ORDER BY tblCustomer.[Customer Name];    Dim strFinalSQL As String    Const sqlCodeBegin = "SELECT tblCustomer.[Customer Number], tblCustomer.[Customer Name] FROM tblCustomer " & \_  "WHERE tblCustomer.Province='"  Const sqlCodeEnd = "' ORDER BY tblCustomer.[Customer Name]"    'Delect the value for the Customer combo box  Me.cboCustomer.Value = Null    'Concatenate the actual selected province with our SQL statement from above  strFinalSQL = sqlCodeBegin & Me.cboProvince.Value & sqlCodeEnd    'Set the rowsource of the customer combo box to the string we just generated  Me.cboCustomer.RowSource = strFinalSQL    'Forces a rerun of the query to ensure the data is up to date.  Me.cboCustomer.Requery    'Me.lblVisits.Caption = strSelectCustomerMsg    Call ClearListBox(Me.lstVisits, Me.lblVisits, strSelectCustomerMsg)  Call ClearListBox(Me.lstLineItem, Me.lblLineItem, strLineItemMsg)    End Sub |

Add the clear list box calls to cboCustomer\_AfterUpdate:

|  |
| --- |
| Private Sub cboCustomer\_AfterUpdate()  Dim strFinalSQL As String  Const strSQLSelect = "SELECT tblPets.[Pet ID], tblPets.[Pet Name]" & \_  "FROM tblPets WHERE tblPets.[Customer Number] = '"  Const strSQLOrder = "' ORDER BY tblPets.[Pet Name]"    strFinalSQL = strSQLSelect & Me.cboCustomer.Value & strSQLOrder  Me.cboPetName.RowSource = strFinalSQL  Me.cboPetName.Requery    'Me.lblVisits.Caption = strSelectPetNameMsg  Call ClearListBox(Me.lstVisits, Me.lblVisits, strSelectPetNameMsg)  Call ClearListBox(Me.lstLineItem, Me.lblLineItem, strLineItemMsg)    End Sub |

Add the clear list box call to cboPetName\_AfterUpdate:

|  |
| --- |
| Private Sub cboPetName\_AfterUpdate()  Dim strFinalSQL As String  Const strSQLSelect = "SELECT tblVisits.[Visit Number], tblVisits.[Pet ID], " & \_  "tblVisits.[Visit Date], tblVisits.[Follow Up Type], tblVisits.[Follow Up Date] " & \_  "FROM tblVisits WHERE tblVisits.[Pet ID] = '"  Const strSQLOrder = "' ORDER BY tblVisits.[Visit Date]"    strFinalSQL = strSQLSelect & Me.cboPetName.Value & strSQLOrder  Me.lstVisits.RowSource = strFinalSQL  Me.lstVisits.Requery  'Me.lblVisits.Caption = "Visits for " & Me.cboPetName.Column(1)    Call ClearListBox(Me.lstLineItem, Me.lblLineItem, strLineItemMsg)  End Sub |

Add it to Form\_Load():

|  |
| --- |
| Private Sub Form\_Load()  'When the form opens all the controls will be cleared and  ' a default caption will be set for the list box    Me.cboProvince.Value = Null  Me.cboCustomer.Value = Null  Me.cboPetName.Value = Null    Me.cboCustomer.RowSource = ""  Me.cboCustomer.Requery    Me.cboPetName.RowSource = ""  Me.cboPetName.Requery    Me.lstVisits.RowSource = ""  Me.lstVisits.Requery    'Me.lblVisits.Caption = strSelectProvinceMsg  Call ClearListBox(Me.lstVisits, Me.lblVisits, strSelectProvinceMsg)  Call ClearListBox(Me.lstLineItem, Me.lblLineItem, strLineItemMsg)  Me.cboProvince.SetFocus    End Sub |

## Add more Features to the list and combo boxes.

There are various properties for list and combo boxes that can only be accessed through VBA code. Some of those are:

* **ListCount** property returns the number of rows in the list box or the list box portion of the combo box.
* **ItemData(index)** property enables you to iterate through the list of entries in a combo or list box.
* **Selected** property sets or returns whether a row is selected.
* **Column** property is more versatile than the ItemData property because it is not restricted to values in the bound column.

**Some examples:**

* **Control.Column(column,row)** We can use the **Column** property to refer to a specific column, or column and row combination, in a multi-column combo box or list box. Use 0 to refer to the first column, use 1 to refer to the second column, etc. Use 0 to refer to the first row, 1 for the second, etc.
* **Control.Selected(itemNumber)** We can use the **Selected** property to select items in a list box. OR you can use the **Selected** property in VBA to determine if an item in a list box is selected (-1 means it is selected and 0 means it is not)
* **Control.ListCount** We can use the **ListCount** property to determine the number of rows in a list box or the list box portion of a combo box. This is useful for displaying the number of records found or iterating through each row in a loop.

**Example**

We will reference the visit selected in the list box by highlighting the large treatment costs in the visit details combo box.

1. Add new features to the **lstVisits\_DblClick** code show below:

|  |
| --- |
| Private Sub lstVisits\_DblClick(Cancel As Integer)  ' When the user double clicks on visit, the line items related to that visit  ' will populate the lstLineItem list box    Dim intCount As Integer  Dim maxCost As Double    'Check to see if the list box contains visits. If it doesn't, they will be  ' double clicking in an "empty" list box    If Me.lstVisits.ListCount <> 0 Then  Call FillLineItemList    'Find the detail with the highest treatment costs and highlight it  intCount = 0    Do While (intCount < Me.lstLineItem.ListCount)  If IsNumeric(Me.lstLineItem.Column(3, intCount)) Then  If CDbl(Me.lstLineItem.Column(3, intCount)) > maxCost Then  Me.lstLineItem.Selected(intCount) = True  maxCost = CDbl(Me.lstLineItem.Column(3, intCount))  End If  End If  intCount = intCount + 1  Loop  End If  End Sub |

# 10.4 Use Advanced Queries to Alter How Data is Displayed in Form Controls

## Union Queries

Over the next few pages, we will alter **frmPetVisits** form in the **KelseyVets.accdb** database to add the option to display all Pets and all Visits.

To accomplish this, we need to create an advanced query using the UNION operator. A UNION is **not a join**, but it is a way of combining data from multiple tables / queries into one display (result set).

Rules for UNION queries:

1. Both tables/queries being combined must have the same number of fields and they must be compatible types.
2. The entire UNION statement can only have one ORDER BY clause.
3. You don’t need to include the DISTINCT modifier because UNION queries don’t return duplicate rows.

## Add an “All” Option to Combo Boxes:

The “(All)” custom item is surrounded with parenthesis because the “(“ (bracket) sorts before numbers and letters making it the first item in the list.

We need to alter:

* The cboPetName combo box Row Source to add the (All) option
* The cboCustomer combo box Row Source to add the (All) option
* The VBA code with dynamic queries that populate the **lstVisit** list box when you select the (All) option from the pet’s combo box.

**Follow these steps:**

* Copy **frmPetVisits** to **frmAllOption**
* Open **frmAllOption**
* Select **cboCustomer**
* Change the Row Source property to the following:
  + SELECT Customer Number, customer name
  + FROM tblCustomer
  + UNION
  + SELECT ‘All’,'(All)'
  + FROM tblCustomer
  + ORDER BY customer name;
* Note that all this code does is add the word (All) to the list. The first SELECT statement has one string field, so the second SELECT statement must also have one string field.

Let’s add the all option to cboCustomer, cboPetNames:

|  |
| --- |
| Private Sub cboProvince\_AfterUpdate()  'Copy and paste the SQL statement into the code. We want to display  ' only customers who are from the province that has been selected.  ' We will need to set the province = to the value of the province combo box    'SELECT tblCustomer.[Customer Number], tblCustomer.[Customer Name] FROM tblCustomer  ' WHERE tblCustomer.Province='AB' ORDER BY tblCustomer.[Customer Name];    Dim strFinalSQL As String    Const sqlCodeBegin = "SELECT tblCustomer.[Customer Number], tblCustomer.[Customer Name] FROM tblCustomer"  Const sqlSecProvStart = " WHERE tblCustomer.Province='"  Const sqlCodeEnd = " UNION SELECT 'All', '(All)' from tblCustomer ORDER BY tblCustomer.[Customer Name]"    'Delect the value for the Customer combo box  Me.cboCustomer.Value = Null    If Me.cboProvince.Value = "All" Then  strFinalSQL = sqlCodeBegin & sqlCodeEnd  Else  strFinalSQL = sqlCodeBegin & sqlSecProvStart & Me.cboProvince.Value & "' " & sqlCodeEnd  End If    'Concatenate the actual selected province with our SQL statement from above  'strFinalSQL = sqlCodeBegin & Me.cboProvince.Value & sqlCodeEnd    'Set the rowsource of the customer combo box to the string we just generated  Me.cboCustomer.RowSource = strFinalSQL    'Forces a rerun of the query to ensure the data is up to date.  Me.cboCustomer.Requery    'Me.lblVisits.Caption = strSelectCustomerMsg    Call ClearListBox(Me.lstVisits, Me.lblVisits, strSelectCustomerMsg)  Call ClearListBox(Me.lstLineItem, Me.lblLineItem, strLineItemMsg)    End Sub |

Adjust the cboCustomer\_AfterUpdate:

|  |
| --- |
| Private Sub cboProvince\_AfterUpdate()  'Copy and paste the SQL statement into the code. We want to display  ' only customers who are from the province that has been selected.  ' We will need to set the province = to the value of the province combo box    'SELECT tblCustomer.[Customer Number], tblCustomer.[Customer Name] FROM tblCustomer  ' WHERE tblCustomer.Province='AB' ORDER BY tblCustomer.[Customer Name];    Dim strFinalSQL As String    Const sqlCodeBegin = "SELECT tblCustomer.[Customer Number], tblCustomer.[Customer Name] FROM tblCustomer"  Const sqlSecProvStart = " WHERE tblCustomer.Province='"  Const sqlCodeEnd = " UNION SELECT 'All', '(All)' from tblCustomer ORDER BY tblCustomer.[Customer Name]"    'Delect the value for the Customer combo box  Me.cboCustomer.Value = Null    If Me.cboProvince.Value = "All" Then  strFinalSQL = sqlCodeBegin & sqlCodeEnd  Else  strFinalSQL = sqlCodeBegin & sqlSecProvStart & Me.cboProvince.Value & "' " & sqlCodeEnd  End If    'Concatenate the actual selected province with our SQL statement from above  'strFinalSQL = sqlCodeBegin & Me.cboProvince.Value & sqlCodeEnd    'Set the rowsource of the customer combo box to the string we just generated  Me.cboCustomer.RowSource = strFinalSQL    'Forces a rerun of the query to ensure the data is up to date.  Me.cboCustomer.Requery    'Me.lblVisits.Caption = strSelectCustomerMsg    Call ClearListBox(Me.lstVisits, Me.lblVisits, strSelectCustomerMsg)  Call ClearListBox(Me.lstLineItem, Me.lblLineItem, strLineItemMsg)    End Sub |

Change the cboCustomer\_AfterUpdate():

|  |
| --- |
| Private Sub cboCustomer\_AfterUpdate()  'This event will call the procedure to fill the list box, lstPet  ' with all corresponding pets  Dim strFinalSQL As String  Const strSQLSelect = "SELECT tblPets.[Pet ID], tblPets.[Pet Name] FROM tblPets "  Const strSpecCustStart = "WHERE tblPets.[Customer Number] = '"  Const strSpecCustEnd = "'"  Const strSQLOrder = " UNION SELECT 'All', '(All)' from tblPets ORDER BY tblPets.[Pet Name]"    If Me.cboCustomer.Value = "All" Then  strFinalSQL = strSQLSelect & strSQLOrder  Else  strFinalSQL = strSQLSelect & strSpecCustStart & Me.cboCustomer.Value & strSpecCustEnd & strSQLOrder  End If    'strFinalSQL = strSQLSelect & Me.cboCustomer.Value & strSQLOrder  Me.cboPetName.RowSource = strFinalSQL  Me.cboPetName.Requery    'Me.lblVisits.Caption = strSelectPetNameMsg  Call ClearListBox(Me.lstVisits, Me.lblVisits, strSelectPetNameMsg)  Call ClearListBox(Me.lstLineItem, Me.lblLineItem, strLineItemMsg)    End Sub |

## Change the FillPetList Procedure to Incorporate the “All” Option

With the addition of being able to display either all **Customers** and a selected **Pet**, or all **Pets** from a selected **Customer**, or all **Customers** and all **Pets** we need to alter both the SQL statement and create a structure that will check for the new options.

**Step 1. Alter the SQL Statement Constants:**

1. Switch to the VBA editor – look at the SQL. Stare at it. So far, it only grabs data from the tblVisits and it only allows you to specify a pet. We want to be able to specify a pet AND a customer (so that we have the option to display all the pets from a specific customer). In order to do this, we need to have a where clause where we can specify the customer number.
2. Look at the relationships diagram. Where can we get the customer number from? What table has a relationship with tblVisits that has a customer number field in it. (tblPets)
3. Copy the query from your VBA code, or if you still have it in record source, that’s fine too.
4. We need to alter this query to include the tblPets and the additional Customer Number where clause.

|  |
| --- |
| Dim strFinalSQL As String  Const strSQLSelect = "SELECT tblVisits.[Visit Number], tblVisits.[Pet ID], " & \_  "tblVisits.[Visit Date], tblVisits.[Follow Up Type], " & \_  "tblVisits.[Follow Up Date], tblPets.[Customer Number] " & \_  "FROM tblPets INNER JOIN tblVisits ON tblPets.[Pet ID] = tblVisits.[Pet ID]"  Const strSQLSpecPet = "tblVisits.[Pet ID]='"  Const strSQLSpecCust = "tblPets.[Customer Number]='"  Const strSQLOrder = " ORDER BY tblVisits.[Visit Date]" |

**Step 2.** Create the structure to check for the “All” option:

Note that we have 4 different SQL statements that can be constructed depending on what is selected in the combo boxes:

1. All **Customer** and all **Pets**
2. All **Customers** and a specific **Pet**
3. All **Pets** from a specific **Customer**
4. A specific **Customer** and a specific **Pet**

To demonstrate the use of an If … Then … Else structure, we will use the following to determine which SQL statement to construct:

1. Remove the single assignment to strFinalSQL and replace it with the following If … Then … Else structure:
2. Save the form and test all options
3. Tests:
   1. Alberta > Animal Kingdom > All
   2. Alberta > Animal Kingdom > Jerry
   3. Alberta > All > All
   4. Alberta > All > Bobo

|  |
| --- |
| Dim strFinalSQL As String  Const strSQLSelect = "SELECT tblVisits.[Visit Number], tblVisits.[Pet ID], " & \_  "tblVisits.[Visit Date], tblVisits.[Follow Up Type], " & \_  "tblVisits.[Follow Up Date], tblPets.[Customer Number] " & \_  "FROM tblPets INNER JOIN tblVisits ON tblPets.[Pet ID] = tblVisits.[Pet ID]"  Const strSQLSpecPet = "tblVisits.[Pet ID]='"  Const strSQLSpecCust = "tblPets.[Customer Number]='"  Const strSQLOrder = " ORDER BY tblVisits.[Visit Date]"    If Me.cboCustomer.Value = "All" Then  If Me.cboPetName.Column(1) = "(All)" Then  strFinalSQL = strSQLSelect & strSQLOrder  Else  strFinalSQL = strSQLSelect & " WHERE " & strSQLSpecPet & Me.cboPetName.Value & "'" & strSQLOrder  End If  Else  If Me.cboPetName.Column(1) = "(All)" Then  strFinalSQL = strSQLSelect & " WHERE " & strSQLSpecCust & Me.cboCustomer.Value & "'" & strSQLOrder  Else  strFinalSQL = strSQLSelect & " WHERE " & strSQLSpecPet & Me.cboPetName.Value & "'" & strSQLOrder  End If  End If |

**Step 3. Alter some controls to implement “All” option on a list box.**

We’ll perform the following edits:

* Remove the All option from the PetName combo box
* Have the controls appear and disappear according to the user’s current path through the form.

First, let’s remove the All option from the PetName combo box:

Alter the cboCustomer\_AfterUpdate():

|  |
| --- |
| Private Sub cboCustomer\_AfterUpdate()  'This event will call the procedure to fill the list box, lstPet  ' with all corresponding pets  Dim strFinalSQL As String  Const strSQLSelect = "SELECT tblPets.[Pet ID], tblPets.[Pet Name] FROM tblPets "  Const strSpecCustStart = "WHERE tblPets.[Customer Number] = '"  Const strSpecCustEnd = "'"  Const strSQLOrder = " ORDER BY tblPets.[Pet Name]"    strFinalSQL = strSQLSelect & strSpecCustStart & Me.cboCustomer.Value & strSpecCustEnd & strSQLOrder      'strFinalSQL = strSQLSelect & Me.cboCustomer.Value & strSQLOrder  Me.cboPetName.RowSource = strFinalSQL  Me.cboPetName.Requery    'Me.lblVisits.Caption = strSelectPetNameMsg  Call ClearListBox(Me.lstVisits, Me.lblVisits, strSelectPetNameMsg)  Call ClearListBox(Me.lstLineItem, Me.lblLineItem, strLineItemMsg)    End Sub |

Next, let’s complete the first list box event so that the second list box has all the option to list all Visits Details.

Modify the Subprocedure FillLineItemList():

|  |
| --- |
| Public Sub FillLineItemList()  'SELECT tblVisitDetails.[Visit Number], tblVisitDetails.[Visit Type], tblTreatments.Treatment, tblVisitDetails.[Treatment Price] AS [Treatment Cost], tblVisitDetails.[Medication Price], FormatCurrency([tblVisitDetails]![Treatment Price]+[tblVisitDetails]![Medication Price]) AS [Total Cost], tblVisits.[Pet ID]  'FROM tblVisits INNER JOIN (tblTreatments INNER JOIN tblVisitDetails ON tblTreatments.[Treatment Code] = tblVisitDetails.[Treatment Code]) ON tblVisits.[Visit Number] = tblVisitDetails.[Visit Number]  'ORDER BY tblVisitDetails.[Visit Number];  Const strSQLSelect = "SELECT \* FROM qryLineItems "  Const strSQLSpecVisit = " tblVisitDetails.[Visit Number] = '"  Const strSQLSpecPet = " tblVisits.[Pet ID] = '"  Const strSQLOrder = "' ORDER BY [Visit Type]"    Dim strFinalSQL As String    If Me.lstVisits.Value = "All" Then  'Filter the query according to Pet ID  strFinalSQL = strSQLSelect & "WHERE " & strSQLSpecPet & Me.cboPetName.Value & strSQLOrder  Else  strFinalSQL = strSQLSelect & "WHERE " & strSQLSpecVisit & Me.lstVisits.Value & strSQLOrder  'Filter the query according to Visit Number  End If      'strFinalSQL = strSQLSelect & "WHERE " & strSQLSpecVisit & Me.lstVisits.Value & strSQLOrder    Me.lstLineItem.RowSource = strFinalSQL  Me.lstLineItem.Requery    Me.lblLineItem.Caption = "Line items for Visit Number " & Me.lstVisits.Value    End Sub |

In terms of UX, let’s make some modifications. Let’s hide some controls until there is some actual data available for them.

When the form loads, maybe the only control available for us is the Province combo box.

When a province is selected, have the Customer combo box become visible.

When a Customer is selected, make the Pet combo box visible.

When a Pet is selected, make the first list box listing the visits visible.

And when a visit is selected (double-clicked), make the visit details list box visible.

We can adjust the visibility of a control via the “visible” property, which takes a True (Visible) or False (Not Visible)

First, let’s edit our Form\_Load procedure, add the following lines to hide the respective controls:

|  |
| --- |
| 'Make our controls not visible  Me.cboCustomer.Visible = False  Me.cboPetName.Visible = False  Me.lstVisits.Visible = False  Me.lstLineItem.Visible = False |

In each of our event procedures, we’ll have to make the next control visible:

|  |  |
| --- | --- |
| **Event** | **Addition (At the end)** |
| cboProvince\_AfterUpdate | Me.cboCustomer.Visible = True |
| cboCustomer\_AfterUpdate | Me.cboPetName.Visible = True |
| cboPetName\_AfterUpdate | Me.lstVisits.Visible = True |
| lstVisits\_DblClick | Me.lstLineItem.Visible = True |

Next, let’s fix the first list box so that if there are no visits, we won’t display the All Option.

Goto your PetName event, and change the following:

|  |
| --- |
| If Me.lstVisits.ListCount = 2 Then  Me.lstVisits.RowSource = ""  Me.lstVisits.Requery  End If |

Why a value of “2”? Since we’re UNION-ing the record set with the All option, we’ll always have at LEAST 1 record, event if the original query returns an empty record set. If we include the **Field Headers** that we’ve turned for that control, that means we’ll have at least **2** records.

Next, let’s do some label adjustments. Let’s delete the label associated with the first list box (make note of the name). Create a new label, and rename it to the old name (lblVisits)

Alter the code in the PetName event:

|  |
| --- |
| Private Sub cboPetName\_AfterUpdate()  'SELECT tblVisits.[Visit Number], tblVisits.[Pet ID], tblVisits.[Visit Date], tblVisits.[Follow Up Type], tblVisits.[Follow Up Date]  'FROM tblVisits  'WHERE tblVisits.[Pet ID] = [Enter a Pet ID]  'ORDER BY tblVisits.[Visit Date];  Dim strFinalSQL As String  Const strSQLSelect = "SELECT tblVisits.[Visit Number], tblVisits.[Pet ID], " & \_  "tblVisits.[Visit Date], tblVisits.[Follow Up Type], " & \_  "tblVisits.[Follow Up Date], tblPets.[Customer Number] " & \_  "FROM tblPets INNER JOIN tblVisits ON tblPets.[Pet ID] = tblVisits.[Pet ID]"  Const strSQLSpecPet = "tblVisits.[Pet ID]='"  Const strSQLSpecCust = "tblPets.[Customer Number]='"  Const strSQLOrder = " UNION SELECT 'All', 'All', '(All)', 'All', 'All', 'All' from tblPets " & \_  "ORDER BY tblVisits.[Visit Date]"    If Me.cboCustomer.Value = "All" Then  If Me.cboPetName.Column(1) = "(All)" Then  strFinalSQL = strSQLSelect & strSQLOrder  Else  strFinalSQL = strSQLSelect & " WHERE " & strSQLSpecPet & Me.cboPetName.Value & "'" & strSQLOrder  End If  Else  If Me.cboPetName.Column(1) = "(All)" Then  strFinalSQL = strSQLSelect & " WHERE " & strSQLSpecCust & Me.cboCustomer.Value & "'" & strSQLOrder  Else  strFinalSQL = strSQLSelect & " WHERE " & strSQLSpecPet & Me.cboPetName.Value & "'" & strSQLOrder  End If  End If      'strFinalSQL = strSQLSelect & Me.cboPetName.Value & strSQLOrder  Me.lstVisits.RowSource = strFinalSQL  Me.lstVisits.Requery  'Me.lblVisits.Caption = "Visits for " & Me.cboPetName.Column(1)    If Me.lstVisits.ListCount = 2 Then  Me.lstVisits.RowSource = ""  Me.lstVisits.Requery  Me.lblVisits.Caption = "No visits have been made for this pet"  Me.lstVisits.Visible = False  Me.lstLineItem.Visible = False  Else  Me.lstVisits.Visible = True  Me.lblVisits.Caption = "Visits for " & Me.cboPetName.Column(1)  End If    Call ClearListBox(Me.lstLineItem, Me.lblLineItem, strLineItemMsg)      End Sub |

Finally, let’s alter the label for the second list box to display the total number of records in the Visit Details.

In the FillLineItemList procedure, alter it to the following:

|  |
| --- |
| Me.lblLineItem.Caption = "Line items for Visit Number " & Me.lstVisits.Value & \_  ", Record Count: " & Me.lstLineItem.ListCount - 1 |

Since the “ListCount” property returns the Field Headers as a record, let’s subtract to get an accurate count.

## Dynamic Reports

One method to dynamically change the data displayed on a report is to change the **Filter** property of the report. A **filter** limits a view of data to specific records without requiring you to alter the design of the underlying report. This is the method we will use. We will allow the user to select values on a form and based on those values we will build a filter that is assigned to the report when it is open.

1. Open the **KelseyVetsVBAReport.accdb** database.
2. Open the report, **rptPetVisitInfo**, in design view
3. Let’s take a look at how Filtering works:
   1. With the Report selected, Click on the Data tab in the Property Sheet
   2. To enable filters, we have to have the following options turned on:
      1. Allow Filters = Yes
      2. Set Filter On Load = Yes
   3. In the Filter property, try the following filters and run the report after each:
      1. [Visit Date] BETWEEN #8/3/1998# AND #8/6/1998#
      2. tblCustomer.[Customer Number] = 'AC001'
      3. tblPets.[Pet ID] = 'AC001-04'

Next, let’s figure out how to run a report from VBA. We have a method called **DoCmd.** Open the Object Browser in VBA and inspect that.

1. Open the Object Browser in VBA
2. Select <All Libraries> or <Access> as the Project/Library
3. Under Classes, select DoCmd
4. Scroll through the Members of DoCmd
5. Select **OpenReport** and click on help. This is the method that we will be using to open a report from a form. We will need the following parameters: **ReportName, View, WhereCondition**. We will use **WhereCondition** instead of **FilterName** because we will be creating our SQL dynamically. The **FilterName** requires the name of a stored query.

Next, examine the exist form.

1. Open the form, **frmCreateSusterPetsReport** in Design View
2. Examine all components and familiarize yourself with all the controls and their names
3. Run the form.
4. The code that hides/shows the tab pages based on the option group selection has already been completed in VBA
5. Switch to the VBA Editor and examine the existing code.

Next, add the code to open a report based on the selection criteria:

1. Open the VBA Editor
2. From the Object List, select **cmdGenReport**
3. By Default, it will add the Click event procedure to the code. This is the event to which we want to add our custom code.
4. Enter the following code to the cmdGenReport\_Click procedure.

|  |
| --- |
| Private Sub cmdGenReport\_Click()  'When the user clicks on the Generate Report Button  ' it will open the report with the appropriate "filter".  ' the filter is selected by determining which radio button the user has selected.    Dim strFilter As String    Select Case Me.fraReportChoices.Value  Case 1  'Generate report based on a visit date range  strFilter = "[Visit Date] BETWEEN #" & Me.txtBegDate.Value & \_  "# AND #" & Me.txtEndDate.Value & "#"  Case 2  'Generate report based on a customer number  strFilter = "tblCustomer.[Customer Number] = '" & Me.cboCustomer.Value & "'"  Case 3  'Generate report based on a specified pet  strFilter = "tblVisits.[Pet ID] = '" & Me.cboPet.Value & "'"  End Select    DoCmd.OpenReport "rptPetVisitInfo", acViewPreview, , strFilter    End Sub |

1. Save the code
2. Run and test the form.