C H A P T E R 7

LOVs AND ALERTS

Chapter Objectives

In this Chapter, you will learn about:

Lists of values (LOVs) and alerts are visual objects with which users can interact. They are different from items in that they appear in their own windows and are not positioned on a canvas.

An LOV is used to present a list of values from which a user can choose to populate items on a form. LOVs can be created manually or by using a wizard. In the Exercises in this Chapter, you will create an LOV and learn about the built-ins that you can use to display it.

Alerts are used to present an important message to the user. Alerts also have buttons so that the user can respond to the message that is being displayed. In the Exercises in this Chapter, you will learn how to create an alert and use the built-ins that you can use to display it.

go to contents

LAB 7.1

LISTS OF VALUES (LOVS)

LAB OBJECTIVES

After this Lab, you will be able to:

- Create LOVs
- Display LOVs

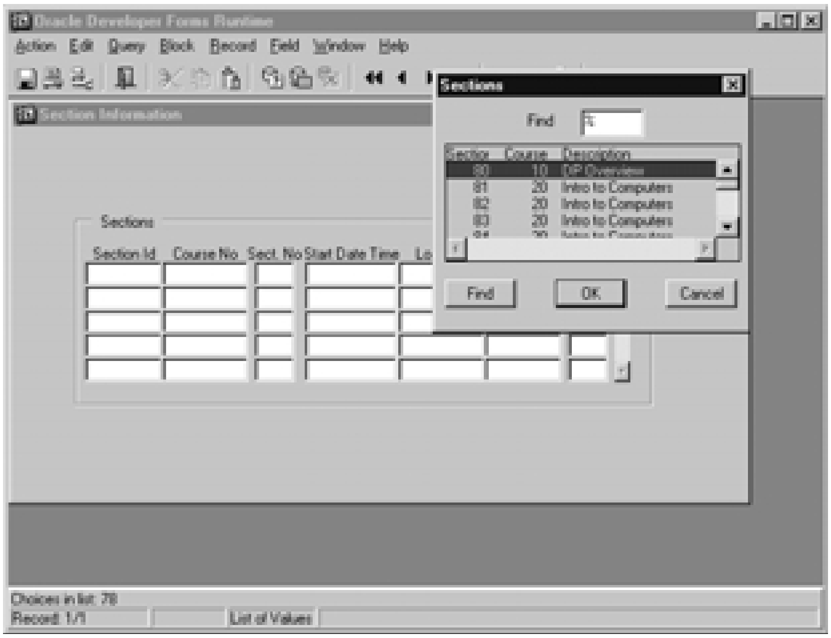
In Chapter 5, "Items," you used list items to present the user with a list of choices. LOVs also present lists of choices to the user, but the lists can be much longer, and they can display multiple columns instead of just one.

■ FOR EXAMPLE:

Figure 7.1 shows an LOV called `Sections`. It has three columns that display the `section_id`, `course_no`, and `description` values. The LOV was displayed when the user entered the `SECTION_ID` item and clicked the `List of Values` button.

To choose an item, the user can select the item in the list and click `OK`, or simply double-click the item. Once selected, some of the column values from the LOV will be returned to the form to populate text and display items. In this case, the `section_id` from the LOV value will be returned to the `SECTION_ID` item in the form.

LOVs serve a number of purposes. They make data entry easier, and they ensure the validity of data. The `SECTION` table has 78 `section_ids`. It would be unreasonable to expect users to memorize all of them along with their corresponding course numbers and descriptions. The LOV provides a quick and easy way for the user to reference the available


Oracle Developer Forms Runtime
Action Edit Query Block Record Field Window Help
Section Information
Sections
Section Id Course No Sect. No Start Date Time
Find
Find OK Cancel
Choices in list: 78
Record: 1/1
List of Values

Section	Course	Description
80	10	DP Overview
81	20	Intro to Computers
82	20	Intro to Computers
83	20	Intro to Computers

Figure 7.1 ■ An LOV showing `section_id`, `course_no`, and `description`.

`section_ids` and then select the one he wants. Not only does this help the user, but it also ensures that the value being put into the text item, and ultimately into the database, is valid. By offering the user the opportunity to choose from a list, you are lessening the chance that the user may enter the data incorrectly.

LOVs are usually assigned/attached to text items. In the example above, the LOV is attached to `SECTION_ID`. When the user navigates to this item, the LOV becomes available and the user can display it and choose from it. In the form shown in Figure 7.1, if the user is not in the `SECTION_ID` item, the LOV is not available and cannot be displayed. However, it is also possible to configure an LOV so that it is not attached to a specific item and is available no matter where the user has navigated. You will explore and experiment with both styles in the Exercises.

All LOVs are based on another Forms object called a record group. Record groups are logical objects, so they have no physical properties and are never displayed to the user. A record group is similar to a database table in that it stores an array of values in a column and row format. A record group can be based on a query or on a set of static values. The `Sections` LOV above was based on a record group that contained the following query:

go to contents

```
SELECT s.section_id, c.course_no, c.description
FROM section s, course c
WHERE c.course_no = s.course_no
ORDER BY section_id
```

In the Exercises, you will use the LOV Wizard to create and configure an LOV. The wizard will also help you create the record group that will serve as the source of your LOV. Once created, you will explore the properties of both objects to further configure them. It is also possible to forego the LOV Wizard and build an LOV manually. Once you have created your LOV, you will experiment with different methods of displaying it.

LAB 7.1 EXERCISES

7.1.1 CREATE LOVS

Open the `R_STUDENT.fmb` form that you created in the "Test Your Thinking" section of Chapter 2, "Wizards and Files." You will use the LOV Wizard to create an LOV for the `STUDENT.ZIP` item. This LOV will display the `ZIP`, `CITY`, and `STATE` values from the `ZIPCODE` table.

a) Based on your experience with the Data Block and Layout Wizards, how do you think you can access the LOV Wizard?

---

b) As you go through and answer the questions on the wizard pages, what kinds of design-time operations will the wizard perform to create your list of values? Do not over-think this question; the answers are rather straightforward.

The LOV Wizard has 9 pages that you will walk through here. The headings in bold indicate the names of the pages according to the Forms help system. The headings will be followed by questions that refer to that page and the function it performs. After each set of questions, there will be a short statement describing how the page should be configured before clicking the `Next` button.

go to contents

Please note that the LOV Wizard has a Welcome and a Finish page that will not be discussed here.

SOURCE PAGE

c) Are there any record group objects in the `R_STUDENT.fmb` form?

d) If there were existing record groups, would you be able to use one of them here as the source of your LOV? How?

Click the `Next` button to continue.

SQL QUERY PAGE

The SQL statement will be the basis of the record group, which will in turn be the source of the LOV. The LOV should display the `ZIP`, `CITY`, and `STATE` columns from the `ZIPCODE` table. The list should be ordered by `STATE` and then `CITY`.

e) What does the `Build SQL Query` button do? Do not use it to build the query, simply click the button to see what it does. Click `Cancel` when you are done.

f) Can you import an existing SQL statement? Where will Forms look for it? Do not import an SQL statement. Simply click the button to see what it does. Click `Cancel` when you are done.

go to contents

g) What should the SQL statement be for this record group? Type it into the `SQL Statement` field.

h) How can you confirm that the SQL you have written is correct?

i) Should you put a semi-colon after the query?

Click the `Next` button to continue.

COLUMN SELECTION PAGE

Again, your LOV should display `ZIP`, `CITY`, and `STATE`.

j) Which columns from the record group should you include in the LOV?

Include the columns and click the `Next` button to continue.

COLUMN DISPLAY PAGE

Adjust the widths of the columns if you'd like, but remember that you can do this later after you have tested the LOV. Adjust the horizontal scrollbar so that you can see the `Return Value` field.

k) When the user makes a selection in the LOV, which of the LOV's columns should be *returned* to the form? `ZIP`, `CITY`, or `STATE`? In this case, *returned* means taken from the LOV and put into an item in the form.

go to contents

l) Which item in which block should this LOV column be returned to?

____

____

m) How does the `Look Up Return Item` button help?

____

____

Make sure the proper return item is specified, and click the `Next` button to continue.

LOV DISPLAY PAGE

There will be no questions for this page. Simply follow the instructions below.

Title the LOV `Zip Codes`. Leave the `Height` and `Width` coordinates as they are. Select `Yes, let Forms position the LOV automatically` and click the `Next` button.

ADVANCED OPTIONS PAGE

There are 227 rows in the `ZIPCODE` table.

n) What would be the difference in behavior in leaving `Retrieve Number of Rows` at `20`, or in switching it to `227`?

____

____

o) How often do state names, city names, and their Zip Codes change? Do you think it is necessary for the form to issue the record group query to refresh the LOV every time it is displayed?

____

____

Leave `Retrieve Number of rows` at `20`. Uncheck `Refresh record group data` before displaying the LOV. Uncheck `Let the user filter records before selecting them`. Click the `Next` button to continue.

go to contents

ITEMS PAGE

So far, you have created a record group and created and configured an LOV.

p) How can you indicate that you want the LOV to be attached to the STUDENT.ZIP item?

Click the Next button to continue to the Finish page. Click the Finish button to return to the Form Builder.

Rename the LOV ZIP_LIST. Run the form and issue a query. Slowly tab from item to item and keep your eye on the Forms Runtime's status line.

q) What happens when you tab into the ZIP item?

Every operating system has a "List of Values" key. When the user presses this key, the LOV will open. On Windows operating systems, it is the F9 key.

r) What happens when you press F9?

Note that the CITY column is a bit too wide and that the STATE column is not visible.

s) What can you do to make the LOV easier to read?

go to contents

7.1.2 Display LOVs

a) When you displayed the LOV in the previous questions, where was it displayed on the screen? Which of the LOV's properties determined that?

Set the LOV's `Automatic Position` property to `No`. Set the LOV's `X Position` and `Y Position` properties to `0, 0`. Run the form, issue a query, and display the LOV.

b) Where is the LOV displayed? Are the X and Y positions relative to the canvas, the window, or the entire screen?

Exit the form and return to the Form Builder.

c) What is the property that will display the LOV as soon as the user navigates to the `STUDENT.ZIP` item? Do not set this property. Simply write its name as your answer.

Set `Auto Skip` to `Yes`. Run the form and issue a query. Tab to the `ZIP` item and display the LOV. Select any Zip Code value from the list and click the `OK` button.

d) Which item did the form navigate to after it closed the LOV? Why might `Auto Skip` be convenient for your users?

go to contents

Look at the properties for the STUDENT.ZIP item and note the ones under the LOV category. Assume that the Validate from List property is set to Yes.

e) What will happen if a user manually enters a value into STUDENT.ZIP that is not in the ZIPCODE table?

So far, you have been displaying the LOV by pressing the F9 key on your keyboard.

f) What type of Forms item could you use to display the LOV?

Create a button and name it SHOW_LIST. Label it List. Make sure it is listed as the last item in the STUDENT block in the Object Navigator. Set its Mouse Navigate property to No. Create a WHEN-BUTTON-PRESSED trigger.

Type the following code into the PL/SQL editor:

```
LIST_VALUES;
```

Run the form and issue a query. Keep the cursor in the STUDENT_ID item. Click the SHOW_LIST button.

g) Was the LOV displayed?

Navigate to ZIP and click the SHOW_LIST button again.

h) Did it work this time? What does this tell you about the way the LIST_VALUES built-in displays LOVs?

Exit the form and return to the Form Builder. Select the WHEN-BUTTON-PRESSED trigger in the Object Navigator and right-click to open the PL/SQL Editor. Erase the code that is there and replace it with the following. Click the Compile button when you are finished.

```
DECLARE
    v_lov boolean;
BEGIN
    v_lov := SHOW_LOV('ZIP_LIST');
END;
```

Run the form and issue a query. Keep the cursor in the STUDENT_ID item. Click the SHOW_LIST button.

i) Was the LOV displayed? What does this tell you about the way the SHOW_LOV built-in displays LOVs?

____

____

LAB 7.1 EXERCISE ANSWERS

7.1.1 ANSWERS

a) Based on your experience with the Data Block and Layout Wizards, how do you think you can access the LOV Wizard?

Answer: You can access the LOV Wizard by selecting Tools | LOV Wizard *from the Main Menu or by right-clicking on any object in the Object Navigator.*

Note that you do not have to select a specific type of object in the Object Navigator to access the LOV Wizard. However, if you want to reenter the wizard for an existing LOV you must select the LOV you wish to edit, then right-click.

b) As you go through and answer the questions on the wizard pages, what kinds of design-time operations will the wizard perform to create your list of values? Do not over-think this question; the answers are rather straightforward.

Answer: The wizard will create the objects that comprise an LOV and will adjust the properties of these objects.

go to contents

As obvious as this seems, it is important to reiterate that the wizard is merely a user-friendly tool that saves you from having to create and configure objects manually. Any object you create, and any properties you configure using any of the wizards, can always be adjusted manually using the Form Builder.

SOURCE PAGE

c) Are there any record group objects in the `R_STUDENT.fmb` form?

Answer: No, because you haven't done anything to create one.

A record group is like a database table in that it is made up of columns and rows. However, record groups belong to a form and are stored locally on a user's machine rather than in a database instance. The group's structure can be defined in three ways:

1) By a query at design-time.
2) Statically at design-time.
3) Programmatically at run-time.

The LOV you create in this Exercise will have a record group based on a query.

d) If there were existing record groups, would you be able to use one of them here as the source of your LOV? How?

Answer: Yes you would by selecting `Existing Record Group` *on the Source Page and choosing a record group from the list item.*

There are two reasons for basing an LOV on an existing record group. First, it allows you to have multiple LOVs based on the same record group. Second, it allows you to base the LOV on a static record group. A static record group is not based on a query. Instead, its source is an array of static values that you define at design-time.

The wizard only creates record groups based on queries; it cannot help you define static record groups. You would have to create and define the static record group manually and then reference it from the LOV Wizard's Source Page.

WHAT THE WIZARD WILL DO ON THIS PAGE: Since you selected `New Record Group based on a query`, the wizard will create a record group object along with the LOV object. The next few wizard pages will define some of the properties for the record group.

go to contents

SQL QUERY PAGE

e) What does the `Build SQL Query` button do? Do not use it to build the query, simply click the button to see what it does. Click `Cancel` when you are done.

Answer: It opens the Query Builder.

The Query Builder is a graphical tool for building SQL statements. It allows you to point and click at tables and their columns to specify the syntax of the SQL statement. It is useful for complicated statements that access multiple tables and columns, or require many joins. It is also helpful for those with poor SQL skills. The Query Builder is unnecessary for this LOV because the SQL statement is rather simple and can be typed in quickly.

f) Can you import an existing SQL statement? Where will Forms look for it? Do not import an SQL statement. Simply click the button to see what it does. Click `Cancel` when you are done.

Answer: Yes you can. Forms will look for it in the filesystem.

g) What should the SQL statement be for this record group? Type it into the `SQL Statement` field.

Answer: The SQL statement should be:

```
SELECT zip, city, state
FROM zipcode
ORDER BY state, city
```

h) How can you confirm that the SQL you have written is correct?

Answer: You can click the `Check Syntax` *button.*

i) Should you put a semi-colon after the query?

Answer: No you should not. If you put a semi-colon after the query, the wizard will return an `Invalid Character, Invalid SQL Statement` *error.*

WHAT THE WIZARD WILL DO ON THIS PAGE: The wizard will take the query you have written here and store it the record group's `Record Group Query` property. You can edit the query later either by reentering the wizard or by adjusting the `SQL Query Statement` property directly for the record group.

COLUMN SELECTION PAGE

Again, your LOV should display ZIP, CITY, and STATE to the users.

j) Which columns from the record group should you include in the LOV?

Answer: You should include all of them since the requirements call for the display of ZIP, CITY, *and* STATE *to the users.*

Why is this page necessary? Why wouldn't you want to display all of the columns you have included in the record group in the LOV? Think back to the Source Page where you were given the option to base an LOV on an existing record group. Essentially, the Source Page is indicating that you can define one query (record group) and use it as the basis for multiple LOVs. There may be columns in the record group that you choose not to display in an LOV.

■ FOR EXAMPLE:

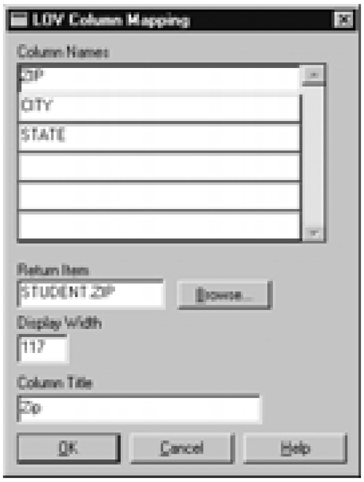
Assume you build a record group called STUDENT_REC based on the following query:

```
SELECT student_id, first_name, last_name
FROM student
ORDER BY last_name;
```

There is a requirement that the application must have two LOVs: one that displays a student's student ID, last name, and first name, and another LOV that displays only the student's last name and first name. The columns for both LOVs exist in the STUDENT_REC record group, so you can use STUDENT_REC for both LOVs. The wizard's Column Selection page displays all of the columns included in the record group and allows you to indicate which columns each LOV should include.

WHAT THE WIZARD WILL DO ON THIS PAGE: The wizard will begin to populate the new LOV's Column Mapping Properties property. This property has multiple values, which are displayed and configured in the LOV Column Mapping dialog shown in Figure 7.2.

This property is one of the most important for an LOV because it not only defines which columns will be displayed, but also each column's return item (which item on the form the LOV column will populate) and the width and title for the LOV column.



Column Names: ZIP, CITY, STATE
Return Item: STUDENT.ZIP
Display Width: 117
Column Title: Zip

Figure 7.2 ■ The `Column Mapping Properties` property's `LOV Column Mapping` dialog.

The wizard configures column display, return items, column width, and the title for an LOV over a series of pages. Once completed, it gathers all of the information you have entered in the separate pages and configures the `Column Mapping Properties` property. If you were to build or edit an LOV manually, you would configure these same settings yourself in the Form Builder by clicking the `Column Mapping Properties` property for the LOV to open the `LOV Column Mappings` dialog.

COLUMN DISPLAY PAGE

k) When the user makes a selection in the LOV, which of the LOV's columns should be *returned* to the form? `ZIP`, `CITY`, or `STATE`? In this case, *returned* means taken from the LOV and put into an item in the form.

Answer: `ZIP` *should be returned to the form.*

l) Which item in which block should this LOV column be returned to?

Answer: `ZIP` *should be returned to the* `STUDENT.ZIP` *item.*

The `Return Value` field specifies which of the LOV columns and their subsequent values should be used to populate items on the form. On this form, the LOV is displayed to give the users a list of Zip Codes to choose from when inserting or updating a student record. So, the LOV's `ZIP` column's value should be returned to the form and it should populate the `STUDENT.ZIP` item.

go to contents

The CITY and STATE columns are included in the LOV so that the Zip Codes make more sense to the user. But, since there are no CITY or STATE values in the STUDENT table and no CITY or STATE items on this form, it is unnecessary to set return item values for these LOV columns. This LOV only has one return item; however, it is possible and common for an LOV to have multiple return items.

m) How does the Look Up Return Item button help?

Answer: It presents a list of available items that can serve as return items.

WHAT THE WIZARD WILL DO ON THIS PAGE: The wizard will continue to populate the LOV's Column Mapping Properties property as it did on the Column Display page.

ADVANCED OPTIONS PAGE

There are 227 rows in the ZIPCODE table.

n) What would be the difference in behavior in leaving Retrieve Number of Rows at 20 or switching it to 227?

Answer: If the Retrieve Number of Rows *value is left at* 20*, Forms will only return 20 rows of the result set at a time. If it is set at* 227*, the entire result set will be returned to the LOV at once.*

The way you should choose to set this value depends on many things, including the number of records in the result set, the amount of traffic on your network, and so on.

A value of 20 could increase the speed at which the LOV gets populated, since only 20 rows will have to be fetched from the database at a time. But, it could degrade overall performance since there will have to be multiple fetches to return the entire result set to the LOV. On the other hand, a value of 227 may cause a delay in the initial population of the LOV because it may take a long time to fetch all of those records. However, it will reduce the amount of fetches required from the database, which could improve performance over the long run. The advantages and disadvantages to either choice depend on many factors.

As the text on the wizard page recommends, it is wise to accept the default for this setting. During the testing phase of your application, you can experiment with different values and consult with your DBA and network administrator to determine the best setting.

o) How often do state names, city names, and their Zip Codes change? Do you think it is necessary for the form to issue the record group query to refresh the LOV every time it is displayed?

Answer: They don't change very often and are unlikely to change during a user's session. Therefore, it is not necessary to refresh the LOV each time it is displayed.

If `Refresh record group data before displaying LOV` is checked, the record group's query will be issued each time the user displays the LOV. This will require resources from the database as well as resulting in increased traffic on the network. The values in the `ZIPCODE` table are fairly static and will not be changed or updated very often. Therefore, it is not necessary to refresh the record group each time the LOV is displayed.

However, what if you had created an LOV on `student_id`, `first_name`, and `last_name`? It is likely that the data in the `STUDENT` table will change during the course of a user's session, so it would be wise to refresh the LOV each time it is displayed despite the database and network resources required.

WHAT THE WIZARD WILL DO ON THIS PAGE: The wizard will set the `Record Group Fetch Size` property for the record group. It will also set the `Automatic Refresh` property for the LOV. It can also set the `Filter Before Display` property for the LOV. If set to `Yes`, this property will present the user with a small filter window to let them filter the values returned to the LOV.

■ *FOR EXAMPLE:*

Assume the user wishes to display the Zip Codes LOV, but they only want to see Zip Codes that begin with 07. If `Filter Before Display` is set to `Yes`, they will be presented with the dialog shown in Figure 7.3. Once they enter their filter criteria, the record group query will be adjusted and then issued. When the LOV opens, it will not include any records that are outside the filter criteria. This will not only make the list more focused for the user, but it will lessen the amount of rows that have to be fetched from the database.

ITEMS PAGE

So far you have created a record group and created and configured an LOV.

p) How can you indicate that you want the LOV to be attached to the `STUDENT.ZIP` item?

Answer: Move `STUDENT.ZIP` *from the* `Return items` *column to the* `Assigned items` *column.*

go to contents

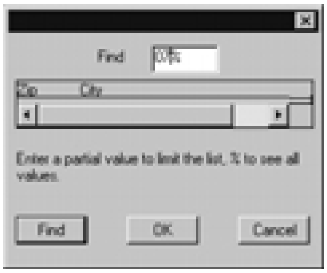

Find
Zip
City
Enter a partial value to limit the list, % to see all values.
Find
OK
Cancel

Figure 7.3 ■ `Filter Before Display` dialog for an LOV.

LOVs are usually attached (or assigned) to the text item or one of the text items that they are to populate. In this case, the Zip Codes LOV is returning a `ZIP` value to the `STUDENT.ZIP` item. It should be attached to the `STUDENT.ZIP` item so that when a user enters this item, the LOV becomes available.

WHAT THE WIZARD WILL DO ON THIS PAGE: The wizard will set the `List of Values` property for the `STUDENT.ZIP` item.

q) What happens when you tab into the `ZIP` item?

Answer: The words "List of Values" appear in the Forms Runtime's status line.

`List of Values` appears on the status line whenever an LOV is available to the current text item.

r) What happens when you press F9?

Answer: The LOV is displayed on the screen.

The LOV has some built-in features, which make it easier for users to work with them. The `Find` field at the top of the LOV lets the user enter criteria to filter the list.

The user can also single-click on one of the items in the list and start typing the value they are looking for. This will start automatically reducing the list to match the value(s) they are typing.

s) What can you do to make the LOV easier to read?

Answer: You can adjust the LOV column widths or make the LOV itself bigger.

Both of these changes can be done either by reentering the wizard or by adjusting the properties directly in the Property Palette. It is probably easiest to make the `CITY` column a bit narrower by setting it to `85` in the Property Palette.

7.1.2 ANSWERS

a) When you displayed the LOV in the previous questions, where was it displayed on the screen? Which of the LOV's properties determined that?

Answer: It was displayed just below `STUDENT.ZIP`*, which is the item it is attached to. The LOV's* `Automatic Position` *property determined this.*

The `Automatic Position` property will relieve you from having to position the LOV yourself. It always positions the LOV just below the item that the LOV is attached to.

b) Where is the LOV displayed? Are the X and Y positions relative to the canvas, the window, or the entire screen?

Answer: It was displayed in the upper right-hand corner. The `X Position` *and* `Y Position` *properties are relative to the entire screen.*

Since the position properties are relative to the entire screen, you can position the LOV outside the canvas and window that contain the current items, or even outside the entire Forms Runtime MDI window. This can be helpful if you don't want the LOV to cover any of the items or objects on the form.

The item that an LOV is attached to also has `LOV X Position` and `LOV Y Position` properties. The item-level position properties override the LOV position properties. This can be helpful if the LOV is reused on multiple forms and you don't want the application to determine the position of the LOV automatically.

c) What is the property that will display the LOV as soon as the user navigates to the `STUDENT.ZIP` item? Do not set this property. Simply write its name as your answer.

Answer: The LOV's `Automatic Display` *property will display the LOV as soon as the user navigates into it.*

Set `Auto Skip` to `Yes`. Run the form and issue a query. Tab to the `ZIP` item and display the LOV. Select any Zip Code value from the list and click the `OK` button.

go to contents

d) Which item did the form navigate to after it closed the LOV? Why might `Auto Skip` be convenient for your users?

Answer: It navigated to the `PHONE` item instead of returning to the `ZIP` item.

This could be helpful to users in that it saves them from having to tab out of the field after they have made their LOV selection. In most cases, you are assuming that after the user has selected from the LOV, she will not have to edit the value that has been returned to the item. Therefore, it is not necessary to return the LOV's item. The form should automatically skip to the next item to speed data entry.

e) What will happen if a user manually enters a value into `STUDENT.ZIP` that is not in the `ZIPCODE` table?

Answer: The LOV will open and the user will not be able to navigate out of the item until a valid value has either been entered or selected from the list.

f) What type of Forms item could you use to display the LOV?

Answer: You could use a button.

g) Was the LOV displayed?

Answer: No it was not.

Navigate to `ZIP` and click the `SHOW_LIST` button again.

h) Did it work this time? What does this tell you about the way the `LIST_VALUES` built-in displays LOVs?

Answer: See discussion below.

You may have already guessed that `LIST_VALUES` is a built-in that is used to display LOVs. In fact, it is the same built-in that was being used when you were pressing the F9 key in the previous questions.

`LIST_VALUES` will only work if there is an LOV attached to the current item. The cursor must be in an item that has an LOV attached to it for `LIST_VALUES` to work. In the instructions for this question, you were asked to set the `SHOW_LIST` button's `Mouse Navigate` property to `No`. Why was this necessary? If the cursor is able to rest on the button, then the button becomes the current item. Is the LOV attached to the `SHOW_LIST` button? No, it is attached to the `ZIP` text item. The `Mouse Navigate` property prevented the cursor from navigating to `SHOW_LIST` and allowed the `LIST_VALUES` built-in to open the LOV.

While this is acceptable behavior, it creates a small problem. The user must use the mouse to click the button to open the LOV. He cannot simply press the ENTER key to open the LOV, which may be more convenient during data entry. You can get around this by adjusting the WHEN-BUTTON-PRESSED trigger slightly.

■ FOR EXAMPLE:

Change the trigger code so that it now looks like this:

```
GO_ITEM('STUDENT.ZIP');
LIST_VALUES;
```

Run the form and issue a query. Tab through the items until the cursor rests on the SHOW_LIST button and press the ENTER key. The LOV should be displayed. The GO_ITEM built-in causes the form to navigate to the STUDENT.ZIP item so that the LIST_VALUES built-in functions properly.

Select the WHEN-BUTTON-PRESSED trigger in the Object Navigator and right-click to open the PL/SQL Editor. Erase the code that is there and replace it with the following code. Click the Compile button when you are finished.

```
DECLARE
    v_lov boolean;
BEGIN
    v_lov := SHOW_LOV('ZIP_LIST');
END;
```

i) Was the LOV displayed? What does this tell you about the way the SHOW_LOV built-in displays LOVs?

Answer: See discussion below.

The SHOW_LOV built-in is also used to display LOVs, but its functionality is slightly different from that of LIST_VALUES. It does not require that the LOV be attached to an item and it allows you to call an LOV explicitly by name. Note that the parameter for SHOW_LOV is ZIP_LIST, the name of the LOV. Another interesting feature is that SHOW_LOV is a Boolean function that will return TRUE if the user picks a value from the LOV and FALSE if they press the LOV's Cancel button. This can be useful to augment processing depending on the user's behavior.

go to contents

■ FOR EXAMPLE:

You can have the form issue messages depending on whether or not the user selects a value from the LOV. Edit the code for the WHEN-BUTTON-PRESSED trigger so that it appears as it does below:

```
DECLARE
    v_lov boolean;
BEGIN
    IF NOT SHOW_LOV('ZIP_LIST')
          THEN MESSAGE ('The user cancelled the
LOV');
    ELSE MESSAGE ('The user selected from the LOV');
    END IF;
END;
```

Run the form and issue a query. Experiment with selecting from the LOV and simply canceling it. Note the different messages that appear on the status line. The commands you put into the THEN and ELSE expressions can be as simple as these or more complicated, depending on how you want the form to react.

LAB 7.1 SELF-REVIEW QUESTIONS

In order to test your progress, you should be able to answer the following questions:

1) What is the difference between an LOV and list item?
- **a)** ____ LOVs open in a separate window, list items are positioned on a canvas
- **b)** ____ LOVs are suited for many rows, list items should be limited to 15 rows
- **c)** ____ LOVs show multiple columns, list items show only one
- **d)** ____ All of the above

2) Upon which Forms object are LOVs based?
- **a)** ____ Block
- **b)** ____ Record group
- **c)** ____ Query
- **d)** ____ Item

3) How can you create an LOV?
- **a)** ____ Manually
- **b)** ____ Using the Layout Wizard
- **c)** ____ Using the LOV Wizard
- **d)** ____ a & c

go to contents

4) How can you size an LOV window?
a) ____ Using the Property Palette
b) ____ Using the Layout Editor
c) ____ Using the LOV Wizard
d) ____ a & c
e) ____ a, b, & c

5) How do you populate a form item with a selection from an LOV?
a) ____ Using a `SELECT...INTO` statement
b) ____ Using a return item
c) ____ You can't
d) ____ Using a record group

6) An LOV must be refreshed each time it is displayed.
a) ____ True
b) ____ False

7) Which of the following built-ins will display an LOV?
a) ____ `LIST_VALUES`
b) ____ `FIND_LOV`
c) ____ `SHOW_LIST`
d) ____ a & c

8) What value will the `SHOW_LOV` built-in return?
a) ____ The value selected from the LOV
b) ____ `TRUE` if the user selects from the LOV, `FALSE` if they cancel the LOV
c) ____ The position of the LOV
d) ____ The LOV's object ID

Quiz answers appear in Appendix A, Section 7.1.

go to contents

LAB 7.2

ALERTS

LAB OBJECTIVES

After this Lab, you will be able to:

- Create and Display Alerts

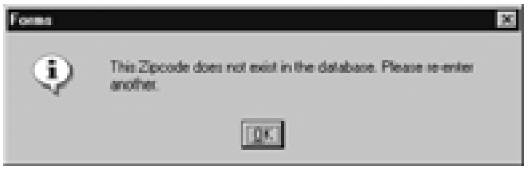
Alerts are windows that contain messages and buttons. They provide a convenient way to send a message to "alert" the user that something has caused an error, something they have done will have certain consequences, or that something has happened that they should simply know about. In previous Exercises and Chapters, you used the `MESSAGE` built-in to send messages to the user. It was convenient and easy, but had two limitations: there was no guarantee that the user would see the message, and there was no way for the user to respond to the message.

Alerts overcome these two limitations in that they appear in the middle of the screen and they require that the user react to them. They can also be configured to let the user make a decision as to how to proceed in response to the alert's message.

■ FOR EXAMPLE:

In Chapter 6, "Triggers & Built-ins" you used the `MESSAGE` built-in to send a message to the user reading, "`This Zip Code does not exist in the database. Please re-enter another.`" There is a chance that the user may miss this type of message since it is a single line of text that appears at the bottom left-hand corner of the MDI window's hint line. You could use an alert instead to display the same message as shown in Figure 7.4.

Alert objects are displayed in the center of the user's screen so they are easy to see. Additionally, the user is not able to return to the form and continue processing until she presses the `OK` button to respond to the

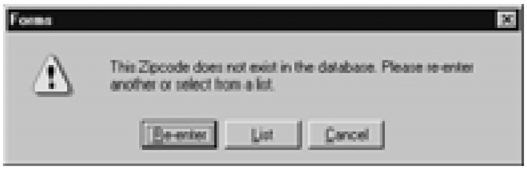


This Zipcode does not exist in the database. Please re-enter another.

Figure 7.4 ■ An alert with one button.

alert. This ensures that the user will have to read and respond to the messages your application is sending them.

Alerts can have up to three buttons. This allows you to present the user with a message along with choices as to how they will respond to that message. So, the alert in Figure 7.4 could be edited slightly to offer choices as shown in Figure 7.5.



This Zipcode does not exist in the database. Please re-enter another or select from a list.

Figure 7.5 ■ An alert with three buttons.

CREATE ALERTS

Alerts are created and configured using the Object Navigator and Property Palette. They are fairly simple objects with only seven Functional properties. You will explore these in the Exercises. Like other visual objects, they have Color and Font properties as well.

Since an alert is a separate window, you do not position it on a canvas. Therefore, it is not visible in the Layout Editor, or anywhere else in the Form Builder at design-time. The only way to *see* an alert is to run the form and display it.

DISPLAY ALERTS

Alerts are displayed using built-ins in much the same way LOVs are displayed with the `SHOW_LOV` built-in. To display an alert, you would use the following built-in function:

go to contents

```
SHOW_ALERT('ALERT_NAME');
```

You can pass the built-in the alert's name, as shown above, or the alert's object ID. SHOW_ALERT is a function that returns a number, so to simply display an alert called ALERT1, the code would be as follows:

```
DECLARE
    v_alert_button   NUMBER;
BEGIN
    v_alert_button := SHOW_ALERT('ALERT1');
END;
```

In the example above, there is no code to respond to the buttons: the alert is simply being displayed. This code could be used to display an alert that has only one button like the one pictured in Figure 7.4. When the button is clicked, the alert will close and control will be returned to the form.

When you create alerts that have more than one button, you need to write code to respond to the button that has been pressed. The SHOW_ALERT built-in returns a number that corresponds to the value of the alert button that was pressed. The three-button alert in Figure 7.5 would require that code be written so that the form could respond differently for each alert button. It would look something like this:

```
DECLARE
    v_alert_button   NUMBER;
BEGIN
    v_alert_button := SHOW_ALERT('ALERT1');
    IF v_alert_button = ALERT_BUTTON1 THEN
          --code to let the user enter another value
    ELSIF v_alert_button := ALERT_BUTTON2 THEN
          --code to display an LOV for zipcodes
    ELSIF v_alert_button := ALERT_BUTTON3 THEN
          --code to cancel
    END IF;
END;
```

The constants ALERT_BUTTON1, ALERT_BUTTON2, and ALERT_BUTTON3 correspond to the buttons in the alert.

As with LOVs, and other objects, you could use the FIND_ALERT built-in to get the alert's object ID.

LAB 7.2 EXERCISES

7.2.1 CREATE AND DISPLAY ALERTS

The purpose of this Exercise is to create an alert, learn how to display it, and learn how to write PL/SQL code to respond to its buttons. You will use form EX07_02.fmb, which has one block based on the ZIPCODE table and an Exit button. The block and its items are not important to this Exercise, since your goal is to focus on alerts.

Open form EX07_02.fmb in the Form Builder. Create an alert in the Object Navigator and name it EXIT_ALERT.

a) Which of the alert's Functional properties have been set by the Form Builder?

b) How do you think you can change the alert's Button Label 2 property so that it will not be displayed to the user?

Change the Message property so that it reads "You pressed the Exit button." Create a WHEN-BUTTON-PRESSED trigger for the ZIPCODE .EXIT button

c) What code could you write behind the WHEN-BUTTON-PRESSED trigger to display this alert? For this question, your code does not have to respond to the alert buttons.

Run the form and click the Exit button. Note the alert's icon. You will need to know it to respond to Question e.

go to contents

d) Can you return to the form without responding to the alert?

Click the alert's `OK` button, then exit the form using the toolbar's `Exit` button to return to the Form Builder.

e) What icon was displayed inside the alert? Which of the alert's properties determined the icon?

Change the `Message` property to "`Are you sure you want to exit the form?`" Change the value of `Button Label 1` to `Yes` and the value of `Button Label 2` to `No`.

f) How should you change the code in the `WHEN-BUTTON-PRESSED` trigger so that the form will exit if the user chooses `Button 1` from the alert? If the user chooses `Button 2`, send a message to the hint line that reads "`You have chosen to continue.`"

Run the form and test the alert. Exit the form and return to the Form Builder. Set the `Default Alert Button` property to `Button 2`. Run the form and display the alert.

g) How did the `Default Alert Button` property affect the alert?

Exit the form and return to the Form Builder.

h) What built-in could you use if you needed to know an alert's object ID? Can you set any of the alert's properties programmatically?

go to contents

i) How could you change this form so that the alert appears whether the user clicks the ZIPCODE.EXIT button or the Exit button on the toolbar? Do not make these changes to the form, simply write your answer below.

LAB 7.2 EXERCISE ANSWERS

7.2.1 ANSWERS

The purpose of this Exercise is to create an alert, learn how to display it, and learn how to write PL/SQL code to respond to its buttons. You will use form EX07_02.fmb, which has one block based on the ZIPCODE table and an Exit button. The block and its items are not important to this Exercise as your goal is to focus on alerts.

Open form EX07_02.fmb in the Form Builder. Create an alert in the Object Navigator and name it EXIT_ALERT.

a) Which of the alert's Functional properties have been set by the Form Builder?

Answer: The Alert Style, Button 1 Label, Button 2, *and* Default Alert Button *properties.*

Alerts are rather simple objects with relatively few properties. The Functional properties define the title, style, buttons, and the message for the alert. You will explore these properties in the rest of the Exercise questions.

The other properties are familiar to you in that they define the color and font of the alert. While it is possible to alter these properties, it is best if you leave them as their default values. A bright red alert with purple text might be more alarming for a user than necessary.

Note that there are no size or positioning properties for an alert. The alert is always displayed in its standard size in the center of the user's screen.

b) How do you think you can change the alert's Button Label 2 property so that it will not be displayed to the user?

Answer: Make the Button Label 2 *property blank.*

As you know, an alert can have as many as three buttons and as few as one. Alert buttons are only displayed if they have a label. By leaving an alert button's label blank, you are indicating to the form that you do not want this button to be displayed.

Button labels can be manipulated programmatically at run-time.

■ FOR EXAMPLE:

To set a button's label to `Yes` at run-time, you would use the following statement:

```
SET_ALERT_BUTTON_PROPERTY('EXIT_ALERT',ALERT_BUTTON1,LABEL'Yes');
```

Change the `Message` property so that it reads "`You pressed the Exit button.`" Create a `WHEN-BUTTON-PRESSED` trigger for the `ZIPCODE.EXIT` button.

c) What code could you write behind the `WHEN-BUTTON-PRESSED` trigger to display this alert? For this question, your code does not have to respond to the alert buttons.

Answer: See the discussion below.

The code would be as follows:

```
DECLARE
    v_alert_button   NUMBER;
BEGIN
    v_alert_button := SHOW_ALERT('EXIT_ALERT');
END;
```

In this example, all you are doing is showing the alert in response to the Button Pressed event. There is no need to evaluate which alert button was pressed since there is only one button on the alert.

This technique is useful if you simply want to display an important message to a user.

■ FOR EXAMPLE:

In Chapter 10, "Reusable Code," you will use an alert to tell the user that validation of an item has failed. No decision will be required of the user, so the alert will have only one button labeled `OK`. The code to call the alert will be similar to the code above since there will only be a need to display the alert, not evaluate any buttons.

Run the form and click the `Exit` button. Note the alert's icon. You will need to know it to respond to Question e.

d) Can you return to the form without responding to the alert?

Answer: No you cannot.

The alert is displayed in what is called a modal window. Modal means that the user cannot leave the window until she has exited it. You will learn more about modality in Chapter 8, "Canvases and Windows." In the case of alerts, users "exit" them by selecting one of their alert buttons. This makes alerts very useful when you want to require users to respond to a question or a message before allowing them to continue.

Click the alert's `OK` button, then exit the form using the toolbar's `Exit` button to return to the Form Builder.

e) What icon was displayed inside the alert? Which of the alert's properties determined the icon?

Answer: A red stop light. The `Alert Style` *property.*

The three alert styles, Stop, Caution, and Note, let you communicate the urgency of an alert's message to the user. Stop alerts should be used for the most urgent messages. Caution alerts can be used for those that are rather urgent and require the user to make a choice. In this Exercise, the message you will present to the user, "`Are you sure you want to exit the form?`", should have `Caution` as its `Alert Style`. Note-style alerts are for those messages that are important, but do not require a response. They will usually only have one button labeled `OK`. The alert you will use in Chapter 10, "Reusable Code," will have `Note` as its `Alert Style`.

Change the `Message` property to "`Are you sure you want to exit the form?`" Change the value of `Button Label 1` to `Yes` and the value of `Button Label 2` to `No`. Change the `Alert Style` to `Caution`.

f) How should you change the code in the `WHEN-BUTTON-PRESSED` trigger so that the form will exit if the user chooses `Button 1` from the alert? If the user chooses `Button 2`, send a message to the hint line that reads "`You have chosen to continue.`"

Answer: See the discussion below.

The code should be as follows:

```
DECLARE
    v_alert_button   NUMBER;
BEGIN
    v_alert_button := SHOW_ALERT('EXIT_ALERT');
    IF v_alert_button = ALERT_BUTTON1 THEN
          EXIT_FORM;
    ELSIF v_alert_button = ALERT_BUTTON2 THEN
          MESSAGE('You have chosen to continue');
    END IF;
END;
```

In this example, you are evaluating the `v_alert_button` variable to determine which button was pressed. The form will exit if the user selects `Button 1` (`Yes`) and will send a message if the user presses `Button 2` (`No`). Sending the message "`You have chosen to continue`" to the user is not really necessary. It was included in this example to show you how the user's response to the alert can be used to initiate other actions. If the alert had three buttons, another `ELSIF` statement would have been included to handle the additional button's response.

Run the form and test the alert. Exit the form and return to the Form Builder. Set the `Default Alert Button` property to `Button 2`. Run the form and display the alert.

g) How did the `Default Alert Button` property affect the alert?

Answer: It displayed the alert with `Button 2` *selected.*

The `Default Alert Button` property determines which button the form should navigate to when the alert is displayed. This can be useful if the users will respond by using the keyboard, and if you think you can predict what their usual response will be.

■ *FOR EXAMPLE:*

If you can predict that your users will normally select `Button 1`, then you should set the `Default Alert Button` property to `Button 1`. That way, when the alert is displayed, the user can simply press `ENTER` to respond to the alert and close it.

Exit the form and return to the Form Builder.

go to contents

h) What built-in could you use if you needed to know an alert's object ID? Can you set any of the alert's properties programmatically?

Answer: You could use `FIND_ALERT` *and* `SET_ALERT_PROPERTY`.

`FIND_ALERT` will find the alert's object ID, which you can then use in the `SHOW_ALERT` built-in or the `SET_ALERT_PROPERTY` built-in. You can use the `SET_ALERT_PROPERTY` built-in to set the title of the alert and the alert's message text programmatically.

FOR EXAMPLE:

The code to set an alert's title and message text at run-time could look like this:

```
DECLARE
    v_alert_id              ALERT;
    v_alert_button    NUMBER;
BEGIN
    v_alert_id := FIND_ALERT('EXIT_ALERT');
    SET_ALERT_PROPERTY(v_alert_id, TITLE, 'Warning');
SET_ALERT_PROPERTY(v_alert_id, ALERT_MESSAGE_TEXT, 'Are you sure you want to exit the form?');
    v_alert_button := SHOW_ALERT(v_alert_id);
    ...
END;
```

Note that, as always, you could also include the `ID_NULL` built-in to evaluate whether or not the alert object exists.

i) How could you change this form so that the alert appears whether the user clicks the `ZIPCODE.EXIT` button or the `Exit` button on the toolbar? Do not make these changes to the form, simply write your answer below.

Answer: You could use a `KEY-EXIT` *trigger to show the alert.*

If you moved the code to show the alert to a `KEY-EXIT` trigger, the alert would be shown whenever the user pressed the Exit key on the keyboard. It would also fire whenever the user clicked the `Exit` button on the toolbar. To get the alert to be shown when the user clicks the `Exit` button on the canvas, you would also have to change the code in the `WHEN-BUTTON-PRESSED` trigger to the following:

```
DO_KEY('EXIT_FORM');
```

This way, when the user clicks the button, it fires the `KEY-EXIT` trigger as well.

go to contents

LAB 7.2 SELF-REVIEW QUESTIONS

In order to test your progress, you should be able to answer the following questions:

1) How is an alert displayed to the user?
a) ____ As a multi-line message on the canvas
b) ____ As a single line of text on the hint line
c) ____ As a separate modal window in the middle of the screen
d) ____ As another forms module

2) How can you position an alert?
a) ____ At design-time using its `X Position` and `Y Position` properties
b) ____ Programmatically at run-time using `SET_ALERT_PROPERTY`
c) ____ The position of an alert cannot be configured
d) ____ a & b

3) How can a user respond to an alert?
a) ____ By clicking one of its buttons
b) ____ By correcting any errors on the form
c) ____ A user cannot respond to an alert
d) ____ By clicking the `OK` button on the toolbar

4) Which of the following statements will display `ALERT1`?
a) ____ `v_alert_button := SHOW_ALERT('ALERT1', 200, 200);`
b) ____ `SHOW_ALERT('ALERT1');`
c) ____ `v_alert_button := SHOW_ALERT('ALERT1');`
d) ____ `FIND_ALERT('ALERT1');`

5) Which of the following statements will correctly evaluate an alert button?
a) ____ `IF v_alert_button = 1 THEN`
b) ____ `IF ALERT_BUTTON1 = 'YES' THEN`
c) ____ `IF v_alert_button = 'YES' THEN`
d) ____ `IF v_alert_button = ALERT_BUTTON1 THEN`

6) How should you configure a simple informational alert?
a) ____ Create a Stop alert with no buttons
b) ____ Create a Caution alert with no message, but one button labeled `OK`
c) ____ Create a Note alert with one button labeled `OK`
d) ____ Create a Stop alert with three buttons

Quiz answers appear in Appendix A, Section 7.2.

go to contents

CHAPTER 7

TEST YOUR THINKING

1) Open the `R_STUDENRL.fmb` file. Create an LOV for the `SECTION_ID` item in the `ENROLL` block. The LOV should display the `section_id` and its corresponding `course_no` and `description`. Return only the `SECTION_ID` to the form. The form should position the LOV automatically. Create a button to display the LOV using the `LIST_VALUES` built-in.

2) Open `the R_INSTRUCTOR.fmb` file. Create an LOV for the `INSTRUCTOR_ID` item in the `INSTRUCTOR` block. The LOV should display the `instructor_id`, `first_name`, and `last_name`, and return `instructor_id` to the form. `last_name` and `first_name` should appear together in one LOV column like this:

```
Smith, Joe
```

The form should position the LOV automatically. Create a button to display the LOV using the `SHOW_LOV` built-in.

go to contents