

Database Case 8

Leedy Rentals Inc.

Problem: Develop a transaction and fee checking system

Management skills: Control
Decide

Access Skills: Forms
Action Queries
Macros

Data Table: LEEDY

Doug Leedy punched the buttons on his hand-held calculator. Three months ago Leedy had been an independent rental car dealer, based in Fargo, North Dakota, when Best Price Rentals, a medium sized operator, offered him a sizeable sum for majority ownership of Leedy Rentals. Best Price had recognized that the recent boom in the Fargo area was merely the start of sustained growth. Best Price also realised that the three existing rental agencies (two of them franchises of national dealerships), would provide intense competition for any start-up enterprise in the area. Best Price decided to approach the single independent dealer with a partnership offer.

Doug was surprised and relieved when the Best Price offer arrived. His 17 year old dealership was straining under the pressure from the newly established national franchises which were able to offer some cut price deals, one-way rentals, and arrangements with frequent flyer schemes which he couldn't. Best Price had a cooperative deal with one of the key domestic airlines and Leedy realised that when rental dealers prices were very similar, customers made decisions based on issues like frequent flyer points. Best Price could also assist during temporary price wars. Faced with deciding between hard times as an independent and receiving cash for losing control, Leedy accepted Best Price's offer.

Leedy soon realized that the extra security offered by Best Price came with additional obligations such as monthly reporting duties. These reports involved the calculations he was currently performing manually on his calculator. Leedy already had a computerized reservation and charging system, a necessity in his industry. However, his system did not have the capability to provide the required reports. Although Best Price were willing to supply their own software, Leedy was happy with his system. He knew that his system did offer the feature of writing details of transactions to an exportable file. With this knowledge, Leedy asked Best Price to send one of their system programmers. Best Price obliged by dispatching Helen Linden.

Helen examined a typical file on Leedy's system and found that it could be readily imported into any common database management system. Although not ideal, this solution would certainly prevent the need for Leedy to manually calculate his reports. Helen advised him to purchase a Windows-based database package, and use it to meet Best Price's monthly reporting requirements.

The management of Best Price needs quarterly information in order to make informed decisions. They require the total revenue derived from each of the car model sizes over each month. Each transaction record contains certain information: a sequential code generated by the computer, date of transaction, model code (S=small, M=medium, L=large, X=sports, W=wagon), car registration, miles travelled, number of rental days, rental fee and a logical field signifying whether the rental was "limited miles" or not. The customer usually has the choice of having unlimited miles or a limit of 100 free miles with a lower base rate. For a limited mile transaction, each mile above 100 miles would incur a per mile cost.

The LEEDY data table supplied on the *Solve it!* disk contains a partial list of transactions downloaded from Leedy Rentals reservation system.

Tasks: There are three tasks in this case:

1. Doug Leedy wants Helen to develop a simple solution for him so that he can check the accuracy of the transactions in the sample list against the paper records he has in his office. He wishes the solution to include an entry screen where the user may enter the transaction code (e.g. S24155). The procedure will then display the record corresponding to the code if it is valid. If the code is not valid the user should be informed the entry is invalid and be required to re-enter the code. The user should be able to exit the system by entering a certain code (e.g. "X"). The entry screen should have ample instructions for the user to use the system.

Hint: use Forms and Form Control Wizards to complete this task.

2. A frustrating aspect of the existing package is that the output files do not contain the rental fee. To meet Leedy's reporting commitments, he must manually calculate the revenue from each transaction. Thus, he is keen for Helen to develop a procedure to calculate the fee for each transaction and place the result in the database. As Best Price requires the total revenue by model type, Leedy wants Helen to include this facility as well. The charges for each model type are listed in the table below:

	Small	Medium	Large	Sports	Wagon
Limited Rate (\$)	45	50	55	60	60
Additional cost per mile (\$)	0.55	0.65	0.75	0.85	0.80
Unlimited Rate (\$)	50	55	60	65	65
Depreciation Rate (%)	3.3	3.2	3.0	3.5	4.4
Purchase Price (\$)	9200	11650	17800	19250	17100

Hint: use a series of Update Action Queries to complete this task.

- *3. Best Price is becoming more demanding in its reporting requirements, adding to Leedy's workload. They wish to know the depreciation of any car in Leedy's yard.

The method of depreciation in the car rental industry is dictated by the Internal Revenue Service. The depreciation amount for each vehicle depends on the total distance travelled, read from the odometer, the rate of depreciation for the model type (e.g. small) and the purchase price of the vehicle. The IRS defines the depreciation amount as a fixed

percentage (i.e. the depreciation rate) of the purchase price, for every 6,000 miles the vehicle has travelled. Develop a procedure to calculate the amount of depreciation for each car based on the depreciation rates and purchase prices listed in the table above. Allow the user to enter the registration of a particular vehicle and then calculate the depreciation on that car.

Hint: use Parameter statements within Queries to complete this task.

Time Estimates (excluding task marked with *):

Expert: 1.5 hour

Intermediate: 2.5 hours

Novice: 4 hours

Tutorial For Database Case 8 Using Access 97



Creating Forms in Access

Forms are used to view, enter or edit data in a database, and are often the most convenient way of performing these operations. Forms can be based on tables or queries, and display data on a record by record basis. As with reports, the forms feature includes a series of wizard dialog boxes, which enable the user to customise the way data will be displayed on a form. With one click of the mouse, you can switch from Form to Datasheet view, which is a tabular view of the same set of records.

With an Access form, you can include lists of values to choose from, use colours to highlight important data, include graphics, and display messages to indicate when an incorrect value has been entered. You can also set up your form so that Access automatically inserts data for you, or prints data on the click of a command button, and/or displays the results of calculations. Let's use the FRIENDS database to create a simple form.


1. Load the practice database FRIENDS.MDB. From the Database window, click on the *Form* object and then click New. From the *New Form* dialog box, click on the  button to the right of the *Choose the Table or Query Where an Object's Data Comes From*, and select the FRIENDS table. Now click and highlight the *Autoform:Columnar* form option which will automatically generate a form that displays fields in a single column. Click OK. Access creates the form and opens it for you. The form shows the first record in your table (see Figure 5-77). Save and name your form (eg: FRIENDS SHOW). In a matter of minutes, you have created a form that can be used for viewing, editing, adding and deleting records in the FRIENDS table.

Figure 5-77

Label displaying text such as a title or caption

Text boxes with attached labels. Text boxes provide an area where you can display or type text or numbers

Check box: indicates a condition. Form Wizards can create check boxes for Yes/No data type fields

Access offers a number of other form wizard types:

Autoform:Datasheet - which displays fields in datasheet format

Autoform:Tabular - which displays each record as a row of fields.

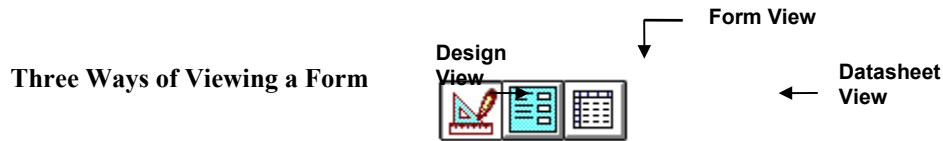
Chart Wizard - creates a form that displays graphs.

Form Wizard - creates a form based on user-selected fields. This wizard supports a variety of form display formats.

Pivot Table Wizard - creates a form with a Microsoft Excel pivot table - a feature which enables analysis and summarization of data in lists and tables.

or you can create your own form from scratch using the *Design View* option. Forms can be also be used with Macros which are discussed later in this tutorial. You may find this useful to know when considering how to tackle some of the tasks associated with the development of the Leedy Rentals system.


There are a number of different control objects which you can use when designing a form. The *Autoform:Columnar* example in Figure 5-xx displays three of these - labels, text boxes and check boxes. Forms use the same set of Toolbox buttons as Reports (recap Access Case 2 tutorial for a listing and short explanation of these buttons).



Access offers three quick and easy ways of viewing a form. You will need to use each of these when designing and using forms. To switch views, simply click on the appropriate toolbar button.

- | | |
|-----------------------|---|
| Design View | use for customising the appearance, or changing the structure of forms, and for adding toolbox controls |
| Form View | use for viewing, entering and editing record data |
| Datasheet View | displays the underlying table or query dynaset on which the form is based |

You can also use the *Print Preview* button to see how a form will look when printed.

Remember: *right click*, means to click the right hand button of your mouse once.
double click top left, means to click the control menu button  on a open window

Creating a Form from Scratch (using the Design View option)

As the name suggests, you start off with a blank form (which is generated by selecting Design View from the New Form dialog box), and use the toolbox buttons in form design view to add all the text boxes, labels and other controls required for your form.. For this reason, blank forms often have no initial connection to an underlying query or table. *Hint: you will need to use the blank form format for Task 1 of Case 8.*

To create a blank form:

- From the Database window, click the Form object and then click New. From the New Form Dialog box, highlight the Design View button. Access immediately opens the form in design view. The initial form will only contain a Detail section. Forms use the same type of sectioning as Reports. Recap Access tutorial for Case 2 for an explanation of report sections.

Let's create a simple customized form using the FRIENDS table. The form is designed to lookup and display specific FRIENDS records based on their ID numbers (Note: Your FRIENDS table will need to contain an ID field set to Autonumber field type). *Hint: the criteria used could just as easily be product numbers or transaction codes.*

- Select VIEW/FORM HEADER/FOOTER from the menu to add a Header to your new form. Next, click on the Label toolbox button, and then click in the Header section of the form. Type a heading for the form (eg: *Friends Lookup*). Change box and font size to suit.



- Click on the Text Box toolbox button, and then click in the **Detail** section of the form to create an unbound text box and label for the form. Click on the label,



and then *right click* and select Properties from the menu. In the Properties window, change the Caption bar to: *Search for a Friend*. Resize your label so that it displays all of the text. *Double click top left* to save and exit back to form design view.

3. Click on the unbound text box, and then *right click* and select Properties (within Properties, select the All Properties tab). Scroll down, click on the Validation Rule bar and enter ≥ 1 And ≤ 16 (eg: where 16 is the last record in your FRIENDS table) to set up a valid ID number range. (Right click and select Zoom if you need more room to type). Click on the Validation Text bar and enter in some infringement text (eg: You don't have that many Friends !! Click OK and ReEnter). *Double click top left* to save and exit back to form design view.

4. Save your form (eg: FRIENDS LOOKUP). Next, we will create two *Command Buttons*. One that links the current form through an event procedure to the FRIENDS SHOW form and displays the relevant records, and one that Exits from FRIENDS SHOW and returns to FRIENDS LOOKUP.

Using Control Wizards in Forms



Creating an *event procedure* in a Report or Form without having to directly do any programming is done through the Control Wizards function. An *event* is a particular action that triggers a procedure such as clicking on a command button in a form or report. A *procedure* is a unit of programming code designed to accomplish a specific task. Access Control wizards automate many frequently used activities associated with forms and reports, such as opening and closing objects, linking fields between objects, and going to specific records in tables or forms.

1. Let's create the first of our command buttons. Click on the Control Wizard toolbox button. Click the *Command Button* tool in the toolbox, and then click on an empty section of the **Detail** section in the FRIENDS LOOKUP form.

Command
Button



The first of several Command Button wizard dialog boxes appears. Make these choices as you proceed through the dialog boxes:

- Under *Categories*, select Form Operations
- Under *Actions*, select Open Form
- Click Next
- Select FRIENDS SHOW as the Form to Open
- Click Next
- Choose to *Open the Form and Find Specific Data to Display*
- Click Next
- From FRIENDS LOOKUP, highlight the Text1 control
- From FRIENDS SHOW, highlight the the ID field and click the <-> button to match the fields
- Click Next
- Select *Text*, and change the button caption to Show Record
- Click Finish to generate the Command Button

2. Save the FRIENDS LOOKUP form again.

3. Right click on the Show Record button and select Properties. Scroll down the list to the On Click: bar. Notice that Access has appended an Event Procedure to this line. Click in the On

Click: bar, and then click on the ... Build button. Access displays the event procedure code behind the Show Record button (see Figure 5-78). *Double click top left* to return to form design view.

Figure 5-78

```
Private Sub Command3_Click()  
On Error GoTo Err_Command3_Click  
  
    Dim stDocName As String  
    Dim stLinkCriteria As String  
  
    stDocName = "Friends Show"  
  
    stLinkCriteria = "[ID]=" & Me![Text1]  
    DoCmd.OpenForm stDocName, , , stLinkCriteria  
  
Exit_Command3_Click:  
Exit Sub  
  
Err_Command3_Click:  
    MsgBox Err.Description  
    Resume Exit_Command3_Click  
  
End Sub
```

4. **Time to Test:** Switch to Form View. Enter an ID number between 1 and 16 (or whatever the last record number in your FRIENDS table is), and click the Show Record Button. The sequence works, but there is no easy or immediate way to run the procedure again, or to exit from FRIENDS SHOW. We need to add an Exit and ReRun button to FRIENDS SHOW.

5. Change to design view in FRIENDS SHOW. Create another command button in the **Detail** section of the form using the control wizards to Exit FRIENDS SHOW and ReRun the ID sequence. This procedure will be based around closing a form. Save the changes to FRIENDS SHOW, return to form view, and test the new button. It should close FRIENDS SHOW and return to FRIENDS LOOKUP.

6. **Extra Practice:** To complete this exercise, we really need a graceful way to exit the FRIENDS LOOKUP procedure and return to the Database window. Create another command button using the control wizard. (*Hint:* it's identical to the one just created in FRIENDS SHOW). Save and test the button, by clicking and exiting to the database window.

7. From the Database window, click Form and FRIENDS LOOKUP and Open to test and run the procedure several times from the beginning. Also enter an ID not in the valid range to make sure the validation procedure is working. With two simple forms, and three easy to generate command buttons, we have created a robust, easy to use FRIENDS lookup program.

Macros in Access

A *macro* in Access is a set of actions that automate common tasks such as opening a table, or printing a report. Macros help you to work smarter and save time without having to learn programming. Macros are simple to create, and Access offers a choice of around 50 different macro actions which include (see also Figure 5-79):

- opening and closing table, forms and reports
- opening a form, and finding records related to another form
- automatically printing reports upon opening a particular database
- checking or improving data validity

Figure 5-79: An Access Macro in Action

You can press a command button on a form ...

Print Monthly Sales

Macro: Print Monthly Sales	
Action	Comment
OpenReport	
Action Arguments	
Report Name	1992 Orders
View	Print
Filter Name	
Where Condition	Datepart("m",[Date])=Datepart("m",Now())

... to run a macro that opens a report. The macro then ...



... finds and prints sales data for the current month.


Source: Access Cue Cards (Access Version 2)

The great thing about macros is that they can handle tasks that would often require extensive programming skills in other database packages, and the more you use them, the more you learn about the underlying principles of programming. Macro actions often involve *arguments*. These are simply parameters which govern how an action is executed. If you have problems choosing macro actions or specifying arguments, press the F1 key for online help.


Let's create a simple macro using the practice database FRIENDS.MDB. This macro will open the FRIENDS table, then display only records whose Zip code is greater than 35000, and then within this, filter out all records where the City is not Arizona, before returning to the full FRIENDS table record set.

1. Load FRIENDS.MDB. Click on the *Table* object in the Database Window, and double click on the FRIENDS table. Change to table design view, and make sure the Zip field is set to a numeric data type. Save necessary changes, and close the FRIENDS table.
2. From the Database Window, click the *Macro* object, and then click *New*. This will display the Macro Builder window. Click in the first blank row of the *Action* column, and then click the button to display a drop down list of all Macro actions. Using your mouse, scroll down this list, and select the *OpenTable* action. (Read the information box at bottom right of the

window, for an explanation of what this action does). In the *Comments* column, type a short description of the action (eg: Opens the FRIENDS table).

3. Click on the *Table Name* bar in the *Action Arguments* section of the window, and then click on the  button to the right of this and select the FRIENDS table.
4. Click on the next empty macro row, and select *ApplyFilter* from the *Action* list. In the *Comments* column, type a short description of the action (eg: Only display records where the Zip code is greater than 35000).
5. Click on the *Where Condition* bar in the *Action Arguments* section and type:

Where Condition	[FRIENDS].[ZIP]>35000
-----------------	-----------------------

Alternatively, click on the expression builder  button to the right of the Where Condition bar to do this.

6. Click on the next empty macro row, and select *ApplyFilter* from the *Action* list. In the *Comments* column, type a short description of the action (eg: Within this reduced record set show records only from Arizona).
7. Click on the *Where Condition* bar in the *Action Arguments* section and type:

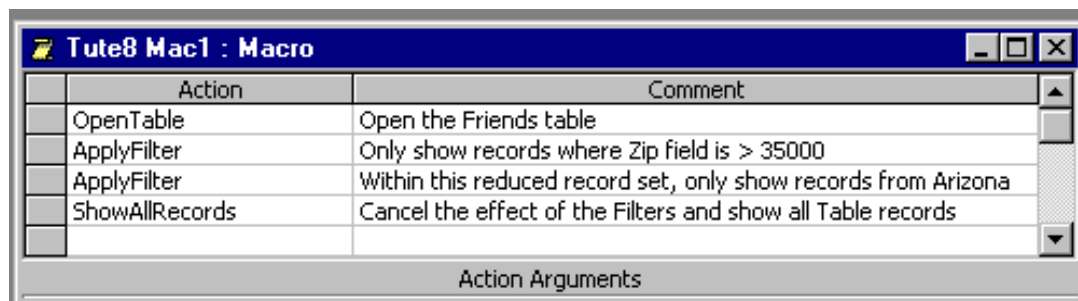
Where Condition	[FRIENDS].[STATE]="Arizona"
-----------------	-----------------------------

or use the expression builder to do this.
(Additional records were added to the

FRIENDS table in the Tutorial for Case 4. If you did not do this tutorial, you will need to add 4 or 5 new records to the FRIENDS table, making sure that some of the state names you enter in the State field duplicate those of existing records. Choose a State name (eg: Arizona) for your macro line that has duplicates).

8. Click on the next empty macro row, and select *ShowAllRecords* from the *Action* list. In the *Comments* column, type a short description of the action (eg: Cancel effect of all Filters). Your macro window should now look similar to Figure 5-80.

Figure 5-80



9. Save your macro by clicking on the *Save* toolbar button or selecting FILE/SAVE from the menu. In the *Save As* window give your macro a name (eg: TUTE8 MAC1), and click *OK*.
10. Execute the macro, by clicking on the *Run* toolbar button, or selecting MACRO/RUN from the menu. As the macro runs, it firstly opens and displays all the records in the FRIENDS table. The first filter is then applied, and only records where the Zip code is greater than 35000 are shown. The second filter then is then applied, and the record set is further reduced to records

where the Zip code is greater than 35000 and the State is Arizona. Finally, the ShowAllRecords action is applied, and the effect of the filters are cancelled. The full record set of the FRIENDS table now appears.

11. Return to the Database window.

Use	• performing tasks such as opening and closing tables or forms, or running reports
Macros	• your application involves custom menus and submenus for forms
When:	• your application is basically simple and uncomplicated, and does not require debugging procedures

Update Queries in Access

Update queries are a type of action query which modifies data according to a specific criteria such as increasing all car rental rates or certain product prices by 15%. Update queries are created in much the same way as other queries except that a new value is specified for a particular field. Like all other types of action queries, update queries save time and effort, but must be used carefully because they actually change the data in the underlying table. Let's use the sample data table HARDWARE to create a simple update query. HARDWARE is included in SOLVE98.MDB.

1. Load the HARDWARE table, and browse the records of the table, particularly noting the data in the Unitcost field (see Figure 5-81).

Table: HARDWARE				
	INVOICE	ITEM	UNITCOST	QUANTITY
▶	1234	Shovel	25	2
	1235	Rake	15	1
	1236	Rack	12.5	4
*				

Figure 5-81
Before Update

Create a new query based on the HARDWARE table. From the query design window, select QUERY/UPDATE from the menu or click on the Update query toolbar button. This action turns the Select query into an Update query, and causes Access to add an *Update to:* line to the QBE Grid.

3. From the Update query design window, select and drag the Item and Unitcost fields in the HARDWARE field list down onto the Field: bar of the QBE Grid. In the *Update to:* cell under the Unitcost field enter the expression [Unitcost]*1.15. This tells Access that we want to increase prices of all items in the HARDWARE table by 15%.

4. Save (eg: TUTE8 QUERY1) and run the query, and then press F11 to return to the Database window. Open the HARDWARE table and notice the changes made to the Unitcost field (see Figure 5-82). Access has updated all prices by 15%.

Table: HARDWARE				
	INVOICE	ITEM	UNITCOST	QUANTITY
▶	1234	Shovel	28.75	2
□	1235	Rake	17.25	1
□	1236	Rack	14.375	4
*				

Update queries can also be limited to certain records or groups of records. Let's say we only wanted to update the unitcost of Rakes. A

simple criteria or *parameter* must be added to the query to impose a filter on the table records. Access offers two ways of doing this:

- In the Criteria: cell under the Item field of the update query window, enter Rake. This tells Access to limit update of the Unitcost field to only those records which contain the word Rake in the Item field.
- alternatively:
- In the Criteria: cell under the Item field of the update query window, enter a parameter statement: [Enter an Item]. Syntax is exactly as shown. When the query is run, Access will display the query Parameter Value window, prompting the user to enter text that will limit the update action (refer Figure 5-83). Entering Rake will produce the same result as method (a).

Figure 5-83

The screenshot shows the 'Update Query: Query1' window. On the left, a table list shows 'HARDWARE' selected. The main area displays a table with columns 'ITEM' and 'UNITCOST'. The 'ITEM' column has a criteria '[Enter the item]' and the 'UNITCOST' column has an update to '[unitcost]*1.15'. A 'Parameter Value' dialog box is open, prompting 'Enter the item' with 'Rake' entered. The dialog has 'OK' and 'Cancel' buttons.

Parameter statements can be used with all types of queries - select and action. They provide an added degree of control by minimising user intervention with the query design window.

Tutorial For Database Case 8 Using Access 2.0




Creating Forms in Access



Forms are used to view, enter or edit data in a database, and are often the most convenient way of performing these operations. Forms can be based on tables or queries, and display data on a record by record basis. As with reports, the forms feature includes a series of wizard dialog boxes, which enable the user to customise the way data will be displayed on a form. With one click of the mouse, you can switch from Form to Datasheet view, which is a tabular view of the same set of records.

With an Access form, you can include lists of values to choose from, use colours to highlight important data, include graphics, and display messages to indicate when an incorrect value has been entered. You can also set up your form so that Access automatically inserts data for you, or prints data on the click of a command button, and/or displays the results of calculations. Let's use the FRIENDS database to create a simple form.

1. Load the practice database FRIENDS.MDB. From the Database window, click on the *Form* object and then click New. From the *New Form* dialog box, click on the  button to the right of the *Select a Table/Query* bar, and select the FRIENDS table. Click on the *Form Wizards* button, and select *Autoform*. Click OK. Autoform creates the form and opens it for you. The form shows the first record in your table (see Figure 5-84). Save and name your form (eg: FRIENDS SHOW)

Note: The same result can be achieved by highlighting the FRIENDS table in the Database Window, and then clicking on the *Autoform* button in the toolbar. In a matter of minutes, you have created a form that can be used for viewing, editing, adding and deleting records in the FRIENDS table.



Figure 5-84

Label displaying text such as a title or caption

Text boxes with attached labels. Text boxes provide an area where you can display or type text or numbers

Check box: indicates a condition. Form Wizards automatically create check boxes for Yes/No data type fields

Autoform bypasses the Form Wizard dialog boxes by automatically creating a form based on a selected table or query. There are four other form wizard types:

Single Column - which displays fields in a single column.

Tabular - which displays each record as a row of fields.

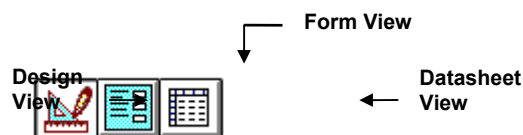
Graph - creates a form that displays graphs.

Main/Subform - creates a form that contains another form. This format displays a one to many relationship between data in the main form (one), and data in the subform (many). Subform data is displayed in datasheet view.

or you can create your own form from scratch. Forms can be also be used with Macros which are discussed later in this tutorial. You may find this useful to know when considering how to tackle some of the tasks associated with the development of the Dickerson Rentals system.

There are a number of different control objects which you can use when designing a form. The Autoform in Figure 5-86 displays three of these - labels, text boxes and check boxes. Forms use the same set of Toolbox buttons as Reports (recap Access Case 2 tutorial for a listing and short explanation of these buttons).


Three Ways of Viewing a Form



Access offers three quick and easy ways of viewing a form. You will need to use each of these when designing and using forms. To switch views, simply click on the appropriate toolbar button.

Design View	use for customising the appearance, or changing the structure of forms, and for adding toolbox controls
Form View	use for viewing, entering and editing record data
Datasheet View	displays the underlying table or query dynaset on which the form is based

You can also use the *Print Preview* button to see how a form will look when printed.

Remember: *right click*, means to click the right hand button of your mouse once.
double click top left, means to click the control menu button  on a open window


Creating a Form from Scratch

As the name suggests, you start off with a blank form, and use the toolbox buttons in form design view to add all the text boxes, labels and other controls required for your form.. For this reason, blank forms usually have no initial connection to an underlying query or table. *Hint: you will need to use the blank form format for Task 1 of Case 8.*

To create a blank form:

1. From the Database window, click the Form object and then click New. From the New Form Dialog box, click on the Blank Form button. Access immediately opens the form in design view. The initial form will only contain a Detail section. Forms use the same type of sectioning as Reports. Recap Access tutorial for Case 2 for an explanation of report sections.

Let's create a simple customised form using the FRIENDS table. The form is designed to lookup and display specific FRIENDS records based on their ID numbers. *Hint: the criteria used could just as easily be product numbers or transaction codes.*

1. In Form design view, select VIEW/PROPERTIES from the menu to open the form properties window. Click on the Record Source bar and set to the FRIENDS table by clicking on the  button and selecting from the drop down list. *Double click top left* in this window to save and return to design view.

2. Select FORMAT/FORM HEADER/FOOTER from the menu to add a Header to your new form. Next, click on the Label toolbox button, and then click in the Header section of the form. Type a heading for the form (eg: *Friends Lookup*). Change box and font size to suit.



3. Click on the Text Box toolbox button, and then click in the Detail section of the form to create an unbound text box and label for the form. Click on the label, and then *right click* and select Properties from the menu. In the Properties window, change the Caption bar to: *Search for a Friend*. Resize your label so that it displays all of the text. *Double click top left* to save and exit back to form design view.



4. Click on the unbound text box, and then *right click* and select Properties. Scroll down, click on the Validation Rule bar and enter ≥ 1 And ≤ 16 (where 16 is the last record in your FRIENDS table) to set up a valid ID number range. (Right click and select Zoom if you need

more room to type). Click on the Validation Text bar and enter in some infringement text (eg: You don't have that many Friends !! Click OK and ReEnter).

5. Save your form (eg: FRIENDS LOOKUP). Next, we will create two *Command Buttons*. One that links the current form through an event procedure to the FRIENDS SHOW form and displays the relevant records, and one that Exits from FRIENDS SHOW and returns to FRIENDS LOOKUP.

Using Control Wizards in



Forms

Control Wizards
Button

Creating an *event procedure* in a Report or Form without having to directly do any programming is done through the Control Wizards function. An *event* is a particular action that triggers a procedure such as clicking on a command button in a form or report. A *procedure* is a unit of programming code designed to accomplish a specific task. Access Control wizards automate many frequently used activities associated with forms and reports, such as opening and closing objects, linking fields between objects, and going to specific records in tables or forms.

1. Let's create the first of our command buttons. Click on the Control Wizard toolbox button, or select VIEW/CONTROL WIZARDS from the menu. Click the *Command Button* tool in the toolbox, and then click on an empty section of the detail section in the FRIENDS LOOKUP form.



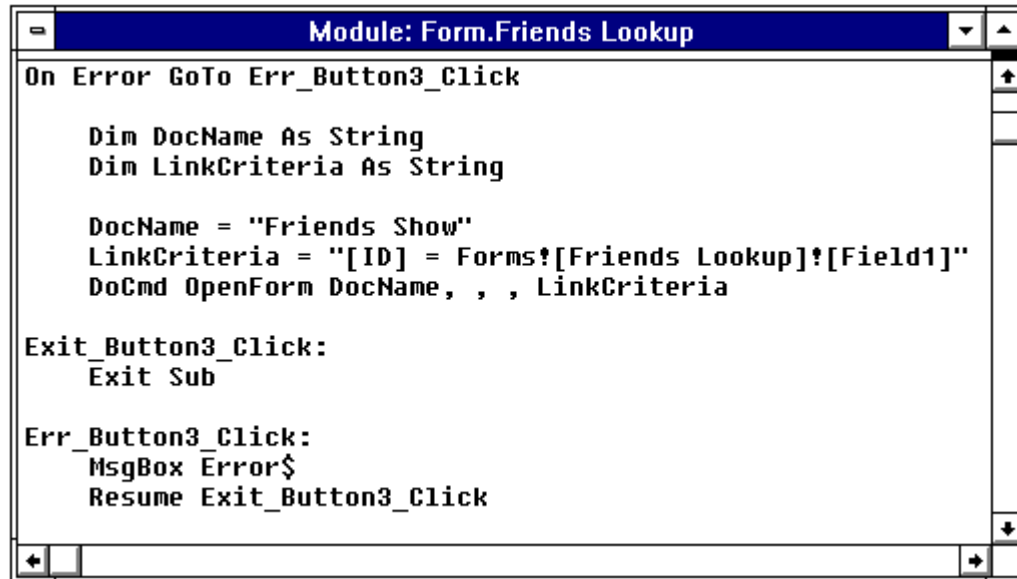
The first of several Command Button wizard dialog boxes appears. Make these choices as you proceed through the dialog boxes:

- Under *Categories*, select Form Operations
- Under *When Button is Pressed*, select Open Form
- Click Next
- Select FRIENDS SHOW as the Form to Open
- Click Next
- Choose to *Open the Form and Find Specific Data*
- Click Next
- From FRIENDS LOOKUP, highlight the Field1 control
- From FRIENDS SHOW, highlight the the ID field and click the <-> button to match the fields
- Click Next
- Select *Text*, and change the button caption to Show Record
- Click Finish to generate the Command Button

2. Save the FRIENDS LOOKUP form again.

3. Click on the Show Record button, and then *right click* and select Properties. Scroll down the list to the On Click: bar. Notice that Access has appended an Event Procedure to this line. Click in the On Click: bar, and then click on the ... Build button. Access displays the event procedure code behind the Show Record button (see Figure 5-84). *Double click top left* to return to form design view.

Figure 5-84



4. **Time to Test:** Switch to Form View. Enter an ID number between 1 and 16 (or whatever the last record number in your FRIENDS table is), and click the Show Record Button. The sequence works, but there is no easy or immediate way to run the procedure again, or to exit from FRIENDS SHOW. We need to add an Exit and ReRun button to FRIENDS SHOW.
5. Change to design view in FRIENDS SHOW. Create another command button using the control wizards to Exit FRIENDS SHOW and ReRun the ID sequence. This procedure will be based around closing a form. Save the changes to FRIENDS SHOW, return to form view, and test the new button. It should close FRIENDS SHOW and return to FRIENDS LOOKUP.
6. **Extra Practice:** To complete this exercise, we really need a graceful way to exit the FRIENDS LOOKUP procedure and return to the Database window. Create another command button using the control wizard. (*Hint:* it's identical to the one just created in FRIENDS SHOW). Save and test the button, by clicking and exiting to the database window.
7. From the Database window, click Form and FRIENDS LOOKUP and Open to test and run the procedure several times from the beginning. Also enter an ID not in the valid range to make sure the validation procedure is working. With two simple forms, and three easy to generate command buttons, we have created a robust, easy to use FRIENDS lookup program.

Macros in Access

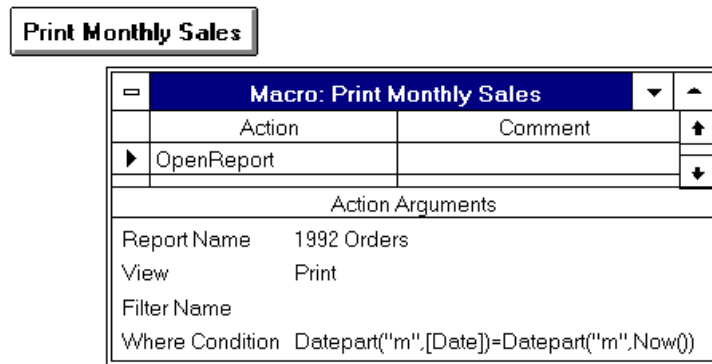
A *macro* in Access is a set of actions that automate common tasks such as opening a table, or printing a report. Macros help you to work smarter and save time without having to learn programming. Macros are simple to create in Access, and this operation is made even simpler if

you use Cue Cards as an interactive assistant. Access offers a choice of 47 different macro actions which include (see also Figure 5-85):

- opening and closing table, forms and reports
- opening a form, and finding records related to another form
- automatically printing reports upon opening a particular database
- checking or improving data validity

Figure 5-85: An Access Macro in Action

You can press a command button on a form ...



... to run a macro that opens a report. The macro then ...

Source: Access Cue Cards



... finds and prints sales data for the current month.

The great thing about macros is that they can handle tasks that would often require extensive programming skills in other database packages, and the more you use them, the more you learn about the underlying principles of programming. Macro actions often involve *arguments*. These are simply parameters which govern how an action is executed. If you have problems choosing macro actions or specifying arguments, press the F1 key for online help and examples screens.

Let's create a simple macro using the practice database FRIENDS.MDB. This macro will open the FRIENDS table, then display only records whose Zip code is greater than 35000, and then within this, filter out all records where the City is not Arizona, before returning to the full FRIENDS table record set.

1. Load FRIENDS.MDB. Click on the *Table* object in the Database Window, and double click on the FRIENDS table. Change to table design view, and make sure the Zip field is set to a numeric data type. Save necessary changes, and close the FRIENDS table.
2. From the Database Window, click the *Macro* object, and then click *New*. This will display the Macro Builder window. Click in the first blank row of the *Action* column, and then click the button to display a drop down list of all Macro actions. Using your mouse, scroll down this list, and select the *OpenTable* action. (Read the information box at bottom right of the window, for an explanation of what this action does). In the *Comments* column, type a short description of the action (eg: Opens the FRIENDS table).
3. Click on the *Table Name* bar in the *Action Arguments* section of the window, and then click on the button to the right of this and select the FRIENDS table.

4. Click on the next empty macro row, and select *ApplyFilter* from the *Action* list. In the *Comments* column, type a short description of the action (eg: Only display records where the Zip code is greater than 35000).

5. Click on the *Where Condition* bar in the *Action Arguments* section and type in the expression:

Where Condition [FRIENDS].[ZIP]>35000

Alternatively, click on the expression builder button to the right of the Where Condition bar to do this.

6. Click on the next empty macro row, and select *ApplyFilter* from the *Action* list. In the *Comments* column, type a short description of the action (eg: Within this reduced record set show records only from Arizona).

7. Click on the *Where Condition* bar in the *Action Arguments* section and type in the expression:

Where Condition [FRIENDS].[STATE]='Arizona'

or use the expression builder to do this.

(Additional records were added to the FRIENDS table in the Tutorial for Case 4. If you did not do this tutorial, you will need to add 4 or 5 new records to the FRIENDS table, making sure that some of the state names you enter in the State field duplicate those of existing records. Choose a State name (eg: Arizona) for your macro line that has duplicates).

8. Click on the next empty macro row, and select *ShowAllRecords* from the *Action* list. In the *Comments* column, type a short description of the action (eg: Cancel effect of all Filters). Your macro window should now look similar to Figure 5-86.

Figure 5-86

Macro: Tute7 Mac1	
Action	Comment
OpenTable	Open the Friends Table
ApplyFilter	Show only those records whose Zip field is greater than 35000
ApplyFilter	Within this reduced record set, show records only from Arizona
ShowAllRecords	Cancel the effect of the Filters and show all Table records
Action Arguments	

9. Save your macro by clicking on the *Save* toolbar button or selecting *FILE/SAVE* from the menu. In the *Save As* window give your macro a name (eg: TUTE8 MAC1), and click *OK*.

10. Execute the macro, by clicking on the *Run* toolbar button, or selecting *MACRO/RUN* from the menu. As the macro runs, it firstly opens and displays all the records in the FRIENDS table. The first filter is then applied, and only records where the Zip code is greater than 35000 are shown. The second filter then is then applied, and the record set is further reduced to records where the Zip code is greater than 35000 and the State is Arizona. Finally, the ShowAllRecords action is applied, and the effect of the filters are cancelled. The full record set of the FRIENDS table now appears.

11. Return to the Database window.

- | | |
|-------------------|--|
| Use Macros | <ul style="list-style-type: none"> performing tasks such as opening and closing tables or forms, or running reports |
| When: | <ul style="list-style-type: none"> your application involves custom menus and submenus for forms your application is basically simple and uncomplicated, and does not require debugging procedures |

Update Queries in Access

Update queries are a type of action query which modifies data according to a specific criteria such as increasing all car rental rates or certain product prices by 15%. Update queries are created in much the same way as other queries except that a new value is specified for a particular field. Like all other types of action queries, update queries save time and effort, but must be used carefully because they actually change the data in the underlying table. Let's use the sample database HARDWARE.MDB to create a simple update query.

1. Load HARDWARE.MDB, and browse the records of the table, particularly noting the data in the Unitcost field (see Figure 5-87).

Figure 5-87 Before Update

Table: HARDWARE				
	INVOICE	ITEM	UNITCOST	QUANTITY
▶	1234	Shovel	25	2
	1235	Rake	15	1
	1236	Rack	12.5	4
*				

2. Create a new query based on the HARDWARE table. From the query design window, select

QUERY/UPDATE from the menu or click on the Update query toolbar button. This action turns the Select query into an Update query, and causes Access to add an *Update to:* line to the QBE Grid.

3. From the Update query design window, select and drag the Item and Unitcost fields in the HARDWARE field list down onto the Field: bar of the QBE Grid. In the *Update to:* cell under the Unitcost field enter the expression [Unitcost]*1.15. This tells Access that we want to increase prices of all items in the HARDWARE table by 15%.

4. Save (eg: TUTE8 QUERY1) and run the query, and then press F11 to return to the Database window. Open the HARDWARE table and notice the changes made to the Unitcost field (see Figure 5-88). Access has updated all prices by 15%.

Figure 5-88 After Update

Table: HARDWARE				
	INVOICE	ITEM	UNITCOST	QUANTITY
▶	1234	Shovel	28.75	2
	1235	Rake	17.25	1
	1236	Rack	14.375	4
*				

Update queries can also be limited to certain records or groups of records. Let's say we only wanted to update the unit cost of Rakes. A simple criteria or *parameter* must be added to the query to impose a filter on the table records. Access offers two ways of doing this:

a) In the Criteria: cell under the Item field of the update query window, enter Rake. This tells Access to limit update of the Unitcost field to only those records which contain the word Rake in the Item field.

alternatively:

b) In the Criteria: cell under the Item field of the update query window, enter a parameter statement: [Enter an Item]. Syntax is exactly as shown. When the query is run, Access will display the query Parameter Value window, prompting the user to enter text that will limit the update action (refer Figure 5-89). Entering Rake will produce the same result as method (a).

Figure 5-89

The screenshot shows the 'Update Query: Query1' window. On the left, a list of tables includes 'HARDWARE'. The main area contains a dialog box titled 'Enter Parameter Value' with the prompt 'Enter the item' and the text 'Rake' entered. Below the dialog box, the query design grid is visible. It shows the 'ITEM' field with the criteria '[Enter the item]' and the 'UNITCOST' field with the update expression '[unitcost]*1.15'.

Parameter statements can be used with all types of queries - select and action. They provide an added degree of control by minimizing user intervention with