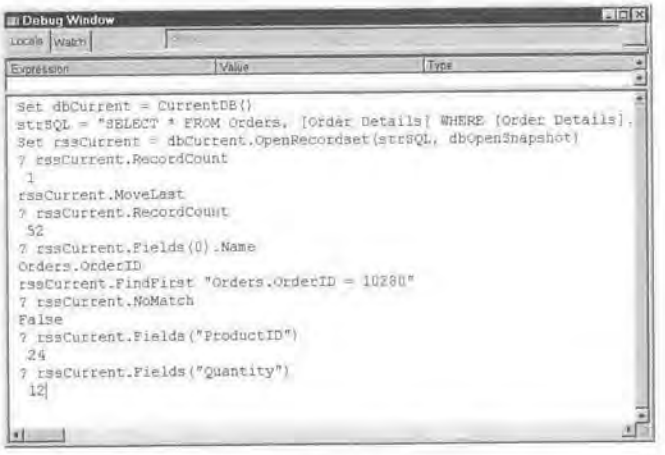


FIG. 29.13 Creating a `Recordset` object with a Jet SQL statement.

Defining a New *QueryDef* Object

You use the `CreateQueryDef` method to create a new, persistent `QueryDef` object and add it to the `QueryDefs` collection. One of the advantages of creating a `QueryDef` is that you can use the name you give the `QueryDef` in place of a table name in a Jet SQL statement. The name of the `QueryDef` must not duplicate the name of an existing `QueryDef` or `TableDef` object. The syntax for creating a new `QueryDef` object is

```
Set qdfName = dbName.CreateQueryDef(strName, strSQL)
```

Tip

You need to apply the `MoveLast` method before testing the value of the `RecordCount` property. If you omit the `MoveLast` method, you usually receive an erroneous `RecordCount` value (1). However, you can test whether any recorders were returned by testing with an `If rstName.RecordCount Then ... End If` structure.

`QueryDef` objects are what their object class name implies: the definition of a query. If `strSQL`'s SQL statement represents an action query, you apply the `Execute` method to the `QueryDef` object to execute the query. You must create a `Recordset` over `QueryDef` objects that define select queries in order to read values in the query result set. Figure 29.14 illustrates how you create a new `QueryDef` object and open a `Recordset` object based on the `QueryDef`. You need to add `Private qdfCurrent As QueryDef` to the Declarations section of your code before you can execute the code shown in the Debug window of Figure 29.14. When you execute a statement that includes the `CreateQueryDef()` method, the new `QueryDef` appears in the Queries page of the Database window. You can open a `QueryDef` created by Access VBA, as shown in Figure 29.15.

Oracle Ex. 1009, p. 1050
Oracle v. DeBellis
IPR2024-00221

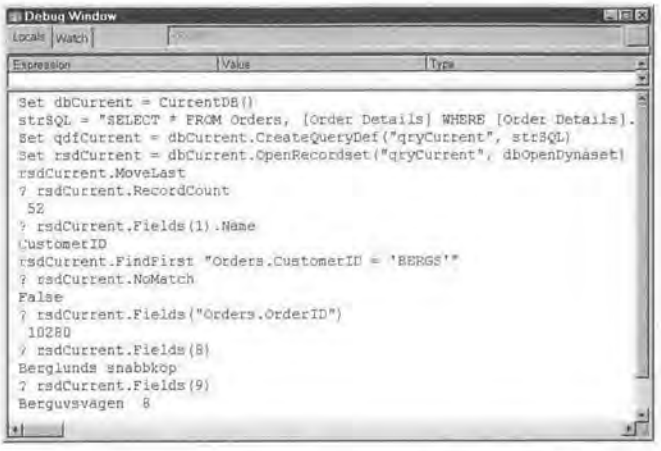


FIG. 29.14 Creating a new `QueryDef` object and opening a `Recordset` over the `QueryDef`.

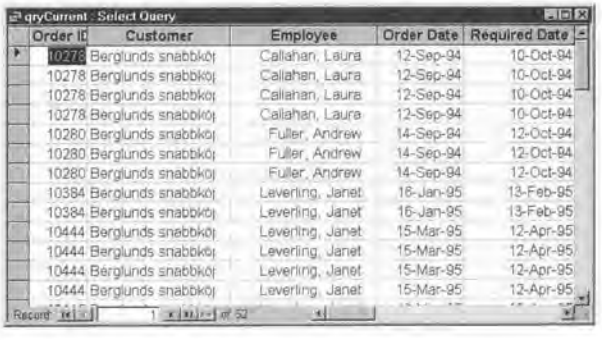


FIG. 29.15 Opening a new `QueryDef` object created with Access VBA code.

Writing a Function that Uses Database Objects

All database objects that you can create and manipulate with Access's graphical development environment also can be created and manipulated with Access VBA code. You can use Access VBA to create a new table, to add records to the table, to define a query to select records from the table, to create a form to display the data, and then to print a report. In other words, you can use Access VBA to create an application exactly the same way you use the dBASE language or PAL. The purpose of Access's graphical development environment, however, is to minimize the need for code in applications.

The `nilTestQuery` function you write in the following example uses the `QueryDef` and `Database` object data types to create a query with a SQL statement and then displays that query in Datasheet view.

Oracle Ex. 1009, p. 1051
Oracle v. DeBellis
IPR2024-00221

The `nilTestQuery` function contains a number of Access VBA reserved words that have been explained in the preceding sections of this chapter, but an example of one reserved word (`DoCmd`) has not yet been provided. Describing every Access VBA reserved word requires a book in itself, as you can see by the length of the Access VBA Language Reference if you have the Office Developer Edition. As you enter the keywords in this example, place the caret inside the keyword and press the F1 key to obtain a detailed explanation of the keyword and its use from Access VBA's Help system.

To create the `nilTestQuery` function, follow these steps:



1. Close any open module; then make the Database window active, click the Modules tab, and click the New button to create a new module.



2. You can accept the default `Option Compare Database` statement that Access adds to the Declarations section of a new module, or you can change the compare modifier to `Text` so that comparisons will not be case-sensitive. Add the `Option Explicit` statement to force explicit declaration of variables if you haven't specified Require Variable Declaration in the options for modules.

3. You need to declare and assign data types to the variables that the procedure uses. Below the **`Option`** `Explicit` statement, type the following statements to create object variables with module-level scope:

```
Private dbNWind As Database
Private qdfTest As QueryDef
```

4. To create the new procedure, type **Function nilTestQuery** below the preceding two statements. It's equally effective to create a subprocedure by typing **Sub TestQuery**.

5. You need to specify the database, in this case Northwind.mdb, that contains the table for the query. Type the following statement:

```
Set dbNWind = CurrentDB()
```

6. You cannot have two `QueryDef` objects of the same name, so you need to delete the `QueryDef` created by multiple executions of the procedure with the *`Collection`*`.Delete` method. Enter the following statements:

```
On Error Resume Next
dbNWind.QueryDefs.Delete "qryTest"
On Error GoTo 0
```

An error is generated if the `QueryDef` you attempt to delete doesn't exist; `qryTest` doesn't exist the first time you run the function, so an error occurs. The **`On Error Resume Next`** statement causes Access to disregard errors that occur in code below the statement. **`On Error GoTo 0`** resumes run-time error checking.

Oracle Ex. 1009, p. 1052
Oracle v. DeBellis
IPR2024-00221

7. You need a variable to hold (point to) the definition of the query. Type the following:

```
Set qdfTest = dbNWind.CreateQueryDef("qryTest")
```

8. Now enter the SQL statement that you use to create and run the query. You can add the SQL statement as the optional second argument of the `CreateQueryDef` method instead of setting the value of its SQL property. Type the following:

```
qdfTest.SQL = "SELECT * FROM Suppliers WHERE SupplierID < 11;"
```

9. Execute the query and display the `Recordset` object created over the `QueryDef` in a datasheet. The Access action, `OpenQuery`, creates a `Recordset` and displays the rows of the `Recordset` in Datasheet view. You use the `DoCmd` object (a reserved word in Access) to execute actions in Access VBA. Enter the following statement:

```
DoCmd.OpenQuery "qryTest"
```

Your Module window appears, as shown in Figure 29.16.

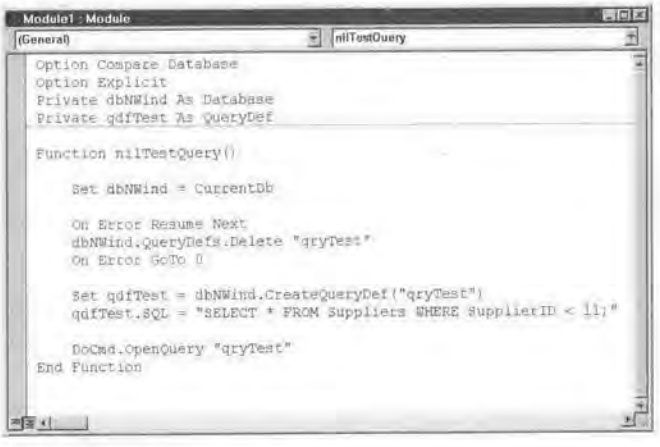


FIG. 29.16 Code for the `nilTestQuery` function.

10. Click the Compile Loaded Modules button of the toolbar or choose Debug, Compile All Modules to verify the syntax of the Access VBA statements you entered.



11. Click the Save button of the toolbar, or choose File, Save, and give the new module a name, such as `modTestQuery`.



12. Open the Debug window and delete any existing entries. Type **? nilTestQuery()**, press Enter, and then close the Debug window. Your query result set appears in the datasheet (see Figure 29.17).



Oracle Ex. 1009, p. 1053
Oracle v. DeBellis
IPR2024-00221

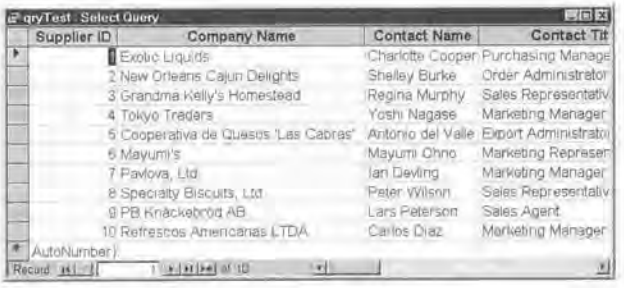


FIG. 29.17 The Datasheet view of the query result set created by the `nilTestQuery` function.

Using ODBCDirect and the Remote Data Object



ODBCDirect is a simplified interface to the RDO. Version 1.0 of the 32-bit (only) RDO was introduced with Visual Basic 4.0 and was included only in the Enterprise Edition. The objective of RDO was to provide a means of using ODBC to connect to client/server RDBMSs without incurring the overhead of the Jet database engine. RDO provides faster processing of SQL pass-through queries, especially for `INSERT` and `UPDATE` operations that involve many records.

In addition, the RDO provides flexible handling of cursors in RDBMSs that implement cursors in their ODBC driver. A *cursor* is a pointer to a row in a query result set, similar to Jet's record pointer. If the RDBMS supports server-side cursors, RDO lets VBA code in your Access 97 front-end manipulate the remote cursor. In situations that involve operations on large query result sets, server-side cursors improve performance and minimize network traffic. Figure 29.18 illustrates the collection hierarchy of the RDO. RDO substitutes the terms *columns* for *fields* and *rows* for *records*, in accordance with conventional RDBMS terminology.



◀◀ See "Using Access 97's SQL Pass-Through Queries," p. 940

Microsoft Office Professional Edition includes RDO 2.0 (Msrdo20.dll), which is required for ODBCDirect, but you can use RDO 2.0 only with ODBCDirect. If you set a reference to Microsoft Remote Data Object 2.0 and attempt to use RDO 2.0 with Access 97, you receive an `ActiveX component can't create object` error. Figure 29.19 compares the collection hierarchy of ODBCDirect and RDO 2.0. Visual Basic developers probably will use RDO 2.0, rather than ODBCDirect, because RDO 2.0 offers additional properties and methods for providing more flexible control of the ODBC connection.

Oracle Ex. 1009, p. 1054
Oracle v. DeBellis
IPR2024-00221

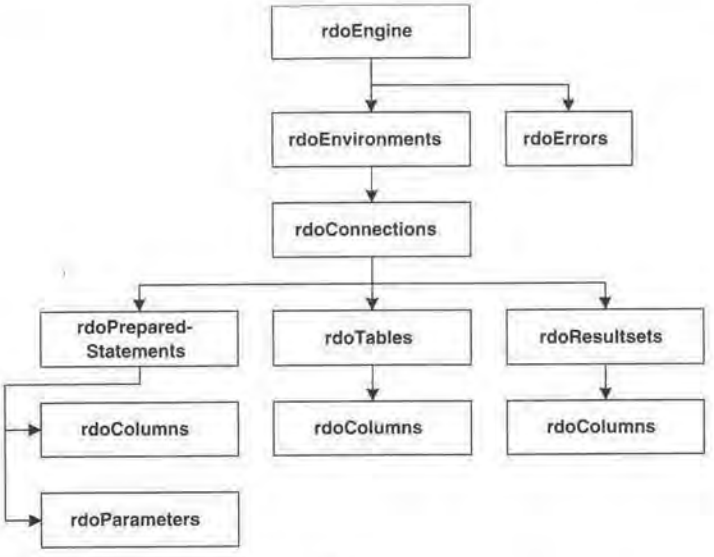


FIG. 29.18 The collection hierarchy of the RDO.

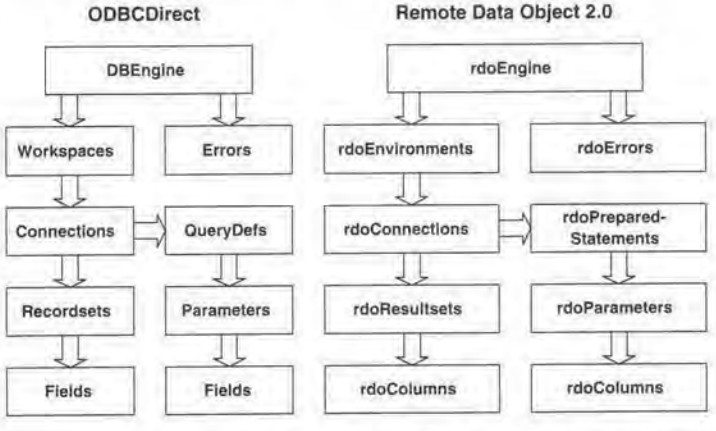


FIG. 29.19 A comparison of the collection hierarchy of ODBCDirect and the RDO.

Visual Basic provides the Remote Data Control (RDC) to bind controls, such as grids and combo boxes, to an `rdoResultset`. Unfortunately, you can't bind `Recordsets` or `QueryDefs` created by ODBCDirect or RDO 2.0 `rdoResultsets` to Access 97 forms, subforms, or controls. This binding limitation, which doesn't apply to `QueryDefs` generated by Jet SQL pass-through queries, greatly diminishes the utility of ODBCDirect and RDO 2.0 in Access applications. Thus, the two sections that follow provide only brief examples of the VBA syntax for ODBCDirect and the RDO.

Oracle Ex. 1009, p. 1055
Oracle v. DeBellis
IPR2024-00221

Note

ODBCDirect and RDO 2.0 offer other advanced features for client/server front-ends, including *asynchronous queries* and input/output parameters for server *stored procedures*. Asynchronous queries let you continue to execute VBA code while the query executes; you also can terminate an asynchronous query before completion. Stored procedures are precompiled queries that reside on the RDBMS server. You can use Jet's SQL pass-through feature to execute stored procedures and send parameter values to stored procedures. ODBCDirect additionally lets you receive data in the form of output parameter values returned by a stored procedure.

Experimenting with ODBCDirect

Creating a `Recordset` object from an ODBCDirect data source involves the following steps:

1. Declare object variables of the `Workspace`, `Connection`, and `Recordset` data type, as in

```
Private wsDirect As Workspace
Private conDirect As Connection
Private rssDirect As Recordset
```

2. Apply the `CreateWorkspace` method to the `DBEngine` object with the `dbUseODBC` constant as the value of the `intType` argument. Specify your SQL Server logon ID and password as the values of the `strUserName` and `strPassword` arguments, as in

```
Set wsDirect = DBEngine.CreateWorkspace("Direct", "sa", "", dbUseODBC)
```

3. Apply the `OpenConnection` method to the `Workspace` object with the name of your ODBC data source as the name of the connection and specify the `dbDriverNoPrompt` option to make the connection to the server without prompting, as in

```
Set conDirect = wsDirect.OpenConnection("Nwind", dbDriverNoPrompt)
```

4. Apply the `OpenRecordset` method to the `Connection` object to create the `Recordset` object, as in

```
Set rssDirect = conDirect.OpenRecordset("Customers")
```

The preceding steps eliminate the need to provide a connection string; ODBCDirect assembles the connection string from the logon ID (`"sa"`) and password (`""`) provided when creating the `Workspace` object, plus the name of the `Connection` object (`"Nwind"`). Your Nwind ODBC data source must specify a database name—nwind, in this example. Invalid entries for any of these three variables result in a run-time error when you use the `dbDriverNoPrompt` option. The default `Recordset` type is a forward-only cursor, which gives the best performance. The `RecordCount` property of a forward-only `Recordset` returns `True` (`-1`) for one or more records and `False` (`0`) if no records are returned. You are limited to the `MoveNext` method to traverse a forward-only `Recordset`.

Oracle Ex. 1009, p. 1056
Oracle v. DeBellis
IPR2024-00221

Note

The online help topics for `CreateWorkspace` and `OpenConnection` provide the full syntax for these methods. Also provided are the names of intrinsic constants for setting the options of the `OpenConnection` method for specifying alternative cursors that let you create bi-directional and updatable `Recordset` objects.

Figure 29.20 illustrates use of the Debug window to experiment with objects based on the ODBCDirect feature of DAO 3.5. The `Connect` property of the `Connection` object returns a complete ODBC connection string, including the database name (`DATABASE=nwind`) obtained from the Nwind ODBC data source. You can use `Recordsets` created over ODBCDirect `Connections` as lookup tables and other operations that don't require binding to Access forms or controls.

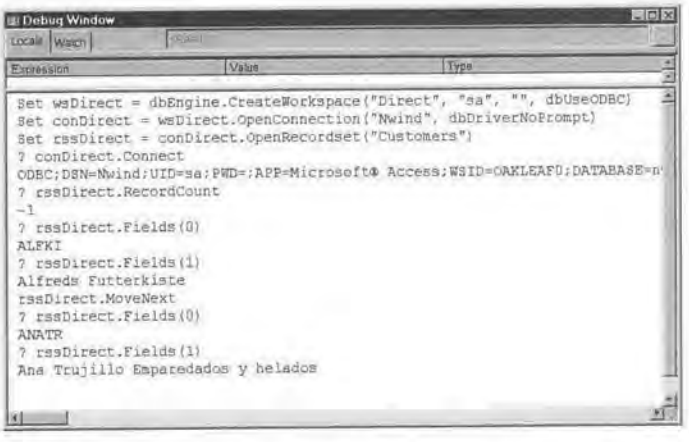


FIG. 29.20 Using the Debug window to create an ODBCDirect `Workspace`, `Connection`, and `Recordset` object.

Oracle Ex. 1009, p. 1057
Oracle v. DeBellis
IPR2024-00221

Chapter 30

Responding to Events with VBA 5.0

All Windows applications are *event-driven*; event-driven means that an event, such as a mouse click on a command button or a change in the position of a record pointer, executes individual blocks of application programming code. Thus the majority of the Access VBA code you write consists of event-handling subprocedures—also called *event procedures*—that are contained within `[{Public|Private}] Sub {Form|Report}_[ObjectName_]EventName ... End Sub` structures of *class modules*. Class module is the new VBA term that replaces Access 2.0's and Access 95's use of *code-behind-forms* (CBF) to describe Access Basic or Access VBA code within a `Form` or `Report` container. This chapter describes how to write Access VBA event-handling code in `Form` and `Report` class modules to automate your Access 97 applications.

Prior versions of Access emphasized the use of Access macros to respond to events. Microsoft promoted Access macros as a simplified programming language for users with little or no programming experience. The repertoire of approximately 40 Access macro actions proved adequate to automate relatively simple applications, and some very large commercial Access applications made extensive use of macros. One of the major drawbacks of Access macros, however, was the inability to handle errors gracefully. Thus, most Access developers abandoned macros in favor of Access 1.x and 2.0, Access Basic, and Access 95 VBA. Now that all of the principal members of Microsoft Office offer VBA, Access macros are on their way to oblivion. There's no guarantee that future versions of Access will continue to support Access macros. Thus, this chapter also shows you how to convert Access macros to VBA code.

Understanding the Role of Class Modules

Class modules, which were introduced by Visual Basic 4.0, are containers for VBA code that relate to a particular class of objects. Access 97 defines two classes (collections)—`Forms` and `Reports`—that contain VBA code for a particular instance of the class: a `Form` or `Report` object, respectively. In object-oriented programming terms, class modules encapsulate VBA code within a

Oracle Ex. 1009, p. 1058
Oracle v. DeBellis
IPR2024-00221

Form or Report object. Code encapsulation lets you create reusable objects. For example, when you copy a form from one Access database to another, the copy you make includes the code in the form's class module.

Access's Form and Report class modules differ from *conventional* modules in that a Form or Report object is integral to the code and contributes the object's properties (appearance). Conventional modules, such as Northwind.mdb's Utility Functions, appear in the Modules page of the Database window. Your event-handling code creates a custom set of methods (behavior) that are applicable to the object. When you open a form or report, you create the *default instance* of the corresponding Form or Report object. The default instance of the object appears in the Forms or Reports page of the Database window. VBA 5.0 also lets you create additional temporary, nondefault instances of Form and Report objects with the **New** reserved word. You need not add an explicit reference to the associated form or report in your code, although in certain expressions you use the **Me** self-reference to specify the current instance of the Form or Report object.

Event-Handling Code in the Main Switchboard Form

The Northwind Traders Main Switchboard form is a good starting point for gaining an understanding of class modules and simple event-handling code. The Main Switchboard form contains two event-handling subprocedures: one each for the Display Database Window and Exit Microsoft Access command buttons, plus a single OpenForm function that services the Categories, Suppliers, Products, Orders, and Print Sales Reports command buttons. Figure 30.1 illustrates the relationships between the command buttons and the function or subprocedure that executes when you click the button.

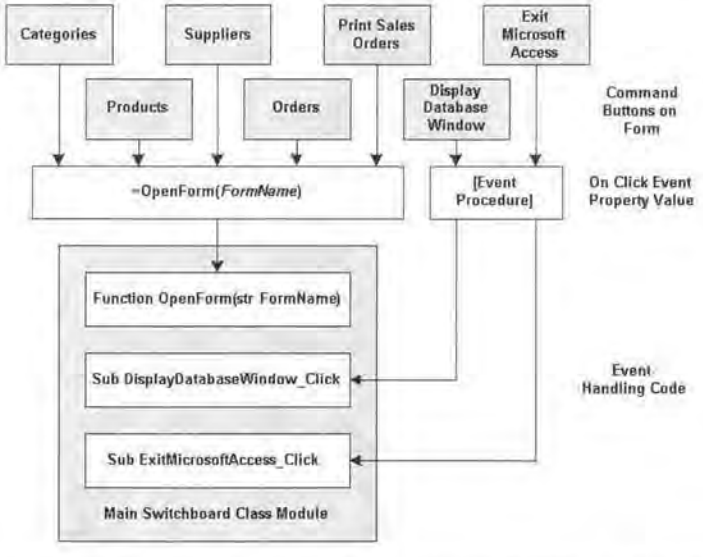


FIG. 30.1 Relationships between command buttons and event-handling code in the Main Switchboard form.

Oracle Ex. 1009, p. 1059
Oracle v. DeBellis
IPR2024-00221

To view the event-handling code in the Main Switchboard form, follow these steps:

1. Open the Main Switchboard form in Design view.
2. Click the Code button of the toolbar to open the Class Module window, Form_Main Switchboard, for the Main Switchboard form. By default, the Class Module window opens with the Declarations section at the top of the window.
3. Open the left drop-down list, which Microsoft calls the Object box, to display a list of the control objects of the form, plus the `Form` object (see Figure 30.2).

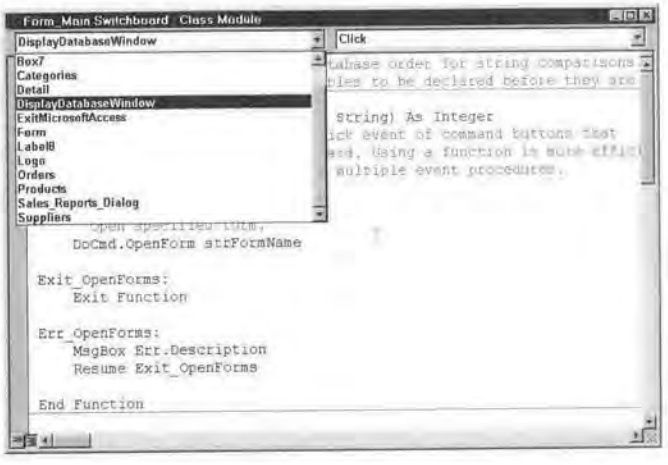


FIG. 30.2 Selecting a form control object in the Object list of the Class Module window.

4. Select one of the control buttons, such as `DisplayDatabaseWindow`, from the Object list to display the subprocedure—**`Sub`** `Form_DisplayDatabaseWindow_Click`—for the On Click event of the Display Database Window command button (see Figure 30.3).

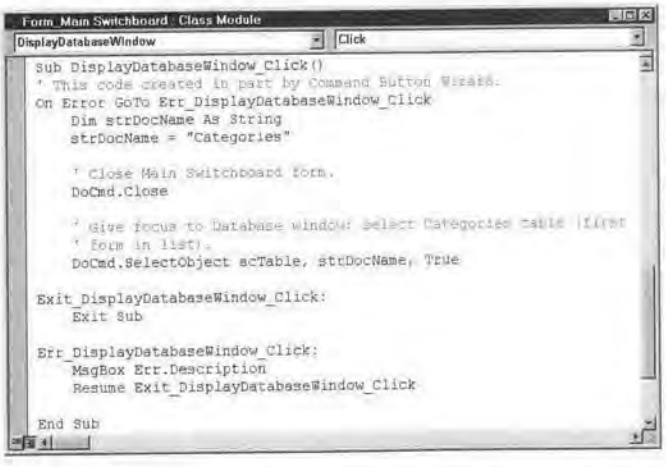


FIG. 30.3 VBA code for the `Form_DisplayDatabaseWindow_Click` event that opens the Database window with the Categories form selected.

Oracle Ex. 1009, p. 1060
Oracle v. DeBellis
IPR2024-00221

5. Open the right drop-down list, which Microsoft calls the Procedure box, to display a list of events applicable to the selection in the Object box. The `Click` event appears in bold type because the `Form_DisplayDatabaseWindows_Click` event subprocedure contains VBA code. When you select an event, such as `DblClick`, without an existing subprocedure, Access creates a *subprocedure stub*. A procedure stub in a class module consists only of the **`Private Sub`** `Form_[ObjectName_]EventName...`**`End Sub`** entries (see Figure 30.4).

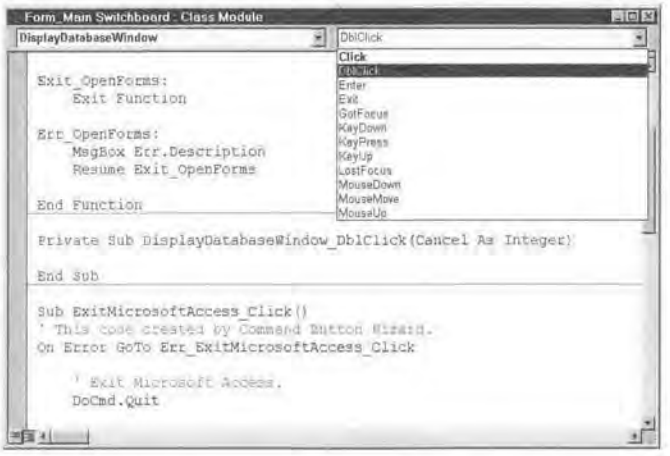


FIG. 30.4 A VBA subprocedure stub created by Access for the `DblClick` event.

Note

All VBA procedures have **`Public`** scope unless you specify otherwise; procedures that you declare with **`Sub`** or **`Public Sub`** are visible to all other class modules and conventional code modules. When you create a procedure stub in a class module, Access adds the **`Private`** modifier. **`Private`** subprocedures and functions have slightly less overhead than **`Public`** subprocedures and function and improve performance in large Access applications. The function and subprocedures of the Form_Main Switchboard class module are declared without the default **`Private`** modifier.

Listing 30.1 shows all of the code contained in the `Form_Main Switchboard` class module. Each of the procedures includes standard error-handling code consisting of **`On Error GoTo`** *`Err_Lable`*`...`**`Resume`** *`Exit_Label`*`...`**`Exit {Function|Sub}`** statements. Adding error-handling to every procedure you write is a generally accepted VBA programming practice. Adding standard error-handling code is one of the subjects of the "Converting Access Macros to VBA Code" section later in the chapter.

◀◀ See "Handling Run-Time Errors," p. 1005

Oracle Ex. 1009, p. 1061
Oracle v. DeBellis
IPR2024-00221

Listing 30.1 Event-Handling Code of the `Form_Main Switchboard` Class Module

```
Option Compare Database  'Use database order for string comparisons.
Option Explicit 'Requires variables to be declared before they are used.

Function OpenForms(strFormName As String) As Integer
'This function is used in the Click event of command buttons that
'open forms on the Main Switchboard. Using a function is more efficient
'than repeating the same code in multiple event procedures.
On Error GoTo Err_OpenForms

    'Open specified form.
    DoCmd.OpenForm strFormName

Exit_OpenForms:
    Exit Function

Err_OpenForms:
    MsgBox Err.Description
    Resume Exit_OpenForms
End Function

Sub ExitMicrosoftAccess_Click()
'This code created by Command Button Wizard.
On Error GoTo Err_ExitMicrosoftAccess_Click

    'Exit Microsoft Access.
    DoCmd.Quit

Exit_ExitMicrosoftAccess_Click:
    Exit Sub

Err_ExitMicrosoftAccess_Click:
    MsgBox Err.Description
    Resume Exit_ExitMicrosoftAccess_Click
End Sub

Sub DisplayDatabaseWindow_Click()
'This code created in part by Command Button Wizard.
On Error GoTo Err_DisplayDatabaseWindow_Click

    Dim strDocName As String
    strDocName = "Categories"

    'Close Main Switchboard form.
    DoCmd.Close

    'Give focus to Database window; select Categories table (first
    'form in list).
    DoCmd.SelectObject acTable, strDocName, True

Exit_DisplayDatabaseWindow_Click:
    Exit Sub
```

(continues)

Oracle Ex. 1009, p. 1062
Oracle v. DeBellis
IPR2024-00221

Listing 30.1 Continued

```
Err_DisplayDatabaseWindow_Click:
    MsgBox Err.Description
    Resume Exit_DisplayDatabaseWindow_Click
End Sub
```

Note

The default Full Module view and Separator Bar settings of the Module page of the Options properties sheet make reading VBA code easier. All of the procedures in the class module appear after the Declarations section of the class module in the alphabetical order of the procedure name, separated by a horizontal gray line. With Full Module view specified, you can use the scroll bars to view all of the procedures within a module.

Access 97's `DoCmd` object, which replaces the **`DoCmd`** statement of Access 2.0 and earlier versions, is the key to manipulating Access application objects with Access VBA. The `DoCmd` statement of Access 1.x and 2.0 used the **`DoCmd`** *`ActionName`*`[`*`Argument(s)`*`]` syntax, where *`ActionName`* was one of approximately 40 predefined macro actions. Application-specific reserved words, such as **`DoCmd`**, preclude a common version of VBA for all members of Office plus Visual Basic; thus `DoCmd` is now an Access-specific object, not a reserved word. The macro actions of Access 2.0 and earlier are now called *methods of the `DoCmd` object*. When you convert an Access 2.0 or earlier database to Access 97, `DoCmd.`*`MethodName`* statements automatically replace **`DoCmd`** *`ActionName`* statements.

Note

The `DoCmd.OpenForm` *`FormName`* statement is the approximate equivalent of Visual Basic's *`FormName`*`.Show [`*`intStyle`*`]` statement. Access's `OpenForm` method, however, has several database-related optional arguments that aren't available with Visual Basic's Show method. Some correspondence takes place between Access's `DoCmd` methods, such as `DoCmd.Beep`, and Visual Basic statements (`Beep`). Architectural differences between Access and Visual Basic, however, preclude identical approaches to programming application-specific objects, such as forms.

Examining Project Class Module Members in the Object Browser. Each `Form` and `Report` object in the current database appears in the Classes list when you select the project name of the current database in the Project/Library (upper) drop-down list Object Browser (see Figure 30.5). By default, the project name for an Access database is the file name of the database without a file extension; thus the project name for Northwind.mdb is Northwind. The default <globals> object displays in the right-hand Members of '<globals>' list all of the procedures in conventional Access modules. These procedures also appear in Members of '*ModuleName*' entries for each module in the project.

Note

To launch the Object Browser, open a form in Design view, and click the Code button of the toolbar to display the class module for the form, then choose View, Object Browser, or press F2.

Oracle Ex. 1009, p. 1063
Oracle v. DeBellis
IPR2024-00221

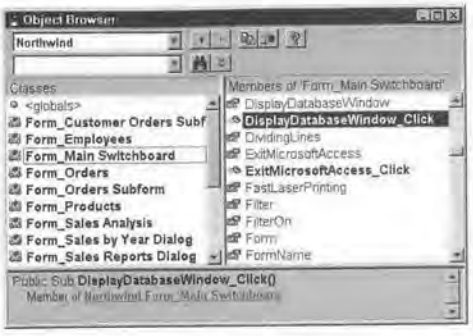


FIG. 30.5 The Object Browser displaying objects in the Northwind project (Northwind.mdb).

> **Note**
>
> You can change the project name of a database by typing a new name in the Project Name text box of the Advanced page of the Options properties sheet.

When you select a `Form` or `Report` object, items representing properties of the `Form` or `Report` object and each of the control objects added to the `Form` or `Report` object appear in the Members of '*ObjectName*' list. Each procedure also appears (in bold type) in the list. Figure 30.5 shows list items for the `DisplayDatabaseWindow_Click` and `ExitMicrosoftAccess_Click` event handlers. If you double-click a procedure item, the Class Module window displays the procedure's code.

> **Note**
>
> VBA 5.0 lets you define your own classes and write custom class modules. Custom class modules give you the opportunity to define a set of properties and methods for the object class you create. Custom class modules appear in Object Browser's Classes list, and the properties and methods you define appear in the Members of '*ObjectName*' list.

Attached add-in libraries, such as the Access Wizards (Wzmain80.mde and Wztool80.mde), appear in the Available References list of the References dialog. You can make objects of attached add-in libraries visible to the Object Browser by adding references to the libraries (see Figure 30.6). As an example, modules in Wzmain80.mde with the AF_ prefix implement the AutoFormat feature of Access 97 (see Figure 30.7). You can start an instance of the AutoFormat feature by typing **? af_Entry** (the entry point function) in the Debug window and pressing Enter. After clicking OK to bypass a few error messages, the AutoFormat dialog appears with an empty Module AutoFormats list. (Close the dialog by clicking the Cancel button.)

> **Note**
>
> To open the References dialog, open a form in Design view, and click the Code button of the toolbar to display the class module for the form, then choose Tools, References.

Oracle Ex. 1009, p. 1064
Oracle v. DeBellis
IPR2024-00221



FIG. 30.6 Creating references to the Access Wizmain80.mde and Wiztool80.mde libraries.



FIG. 30.7 Object Browser displaying conventional modules and members of the <globals> object of the Wzmain80 library.

Adding Event-Handling Code with the Command Button Wizard. Event-handling subprocedures represent the most common approach to handling events that are generated by control objects and `Recordset` objects bound to forms and reports. Command buttons are the most common control object to initiate user-generated events. The easiest way to create a simple VBA event-handling subprocedure is to add a command button to a form with the Command Button Wizard. The Command Button Wizard writes most or all of the code for the most commonly used `Click` event handlers.

To add a command button and its associated event-handling code for opening the Categories form, follow these steps:





1. Open Northwind.mdb, if necessary. Click the Forms tab, and then click the New button to open the New Form dialog.



2. Double-click Design View to open the form in Design view. Click the Toolbox button of the toolbar, if necessary, to display the Toolbox.



3. Make sure that the Control Wizard toggle button is depressed, then click the Command Button tool and add a small button on the form. The first dialog of the Command Button Wizard appears.



Oracle Ex. 1009, p. 1065
Oracle v. DeBellis
IPR2024-00221

4. Select Form Operations in the Categories list and Open Form in the Actions list (see Figure 30.8). Selecting Form Operations displays a sample button with a small form icon. Click Next to continue.

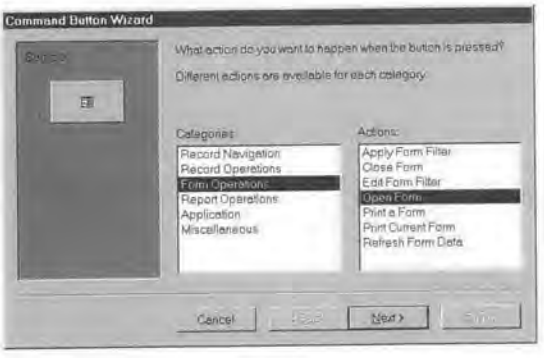


FIG. 30.8 Selecting the category and action for the command button in the first Command Button Wizard dialog.

5. Select Categories (or another form) in the What Form Would You Like To Open list (see Figure 30.9), and click Next to continue.



FIG. 30.9 Selecting the name of the form to open.

6. If the form to open is bound to a table or query, you can allow the form to display all records or add a filter to display only a single record based on the value of a field in your new form. For this example, select the Open the Form and Show All the Records option (see Figure 30.10), and click Next to continue.

7. You can select from a variety of bitmapped icons for the button by marking the Show All Pictures check box. Alternatively, select the Text option button and type a caption for the command button in the text box (see Figure 30.11). Click Next to continue.

8. Type a name for the command button, such as **cmdOpenCategories**, in the text box (see Figure 30.12). The `cmd` prefix is the naming convention for command buttons. Click Finish to add the event-handling code to the class module for the new form and close the last Command Button Wizard dialog.

Oracle Ex. 1009, p. 1066
Oracle v. DeBellis
IPR2024-00221



9. Click the Code button of the toolbar to display the class module for your new form. Figure 30.13 shows the subprocedure added to the module by the Control Wizard. Close the Class Module window to return to your new form in Design view.

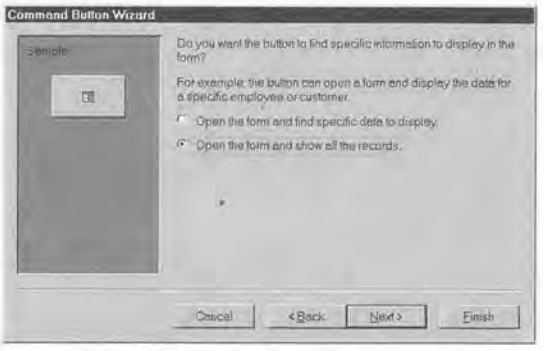


FIG. 30.10 Selecting whether to apply a record filter or allow the opening form to display all records.

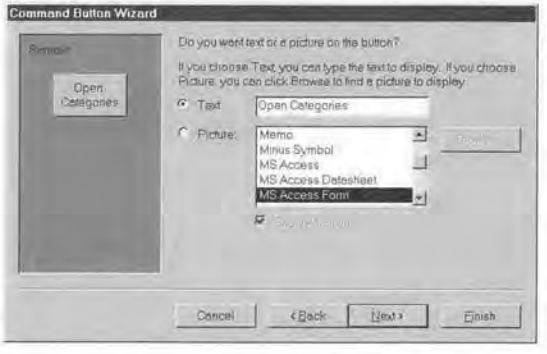


FIG. 30.11 Adding a text caption to the command button.



The event-handling subprocedure you created in the preceding steps is bound to the On Click event of the cmdOpenCategories button. Select the command button, click the Properties button of the toolbar to display the properties sheet for the button, and click the event tab. The [Event Procedure] entry for the On Click event specifies the `cmdOpenCategories_Click` event handler. When you open the drop-down list for the On Click event, [Event Procedure] and the names of all of the macros, if any, in your database appear (see Figure 30.14).



Note

You can create a simple event-handling subprocedure stub for any event of an existing control by clicking the builder button for the event and then double-clicking the Code Builder item in the Choose Builder list. In this case, it's up to you to fill in the code to handle the event.

Oracle Ex. 1009, p. 1067
Oracle v. DeBellis
IPR2024-00221

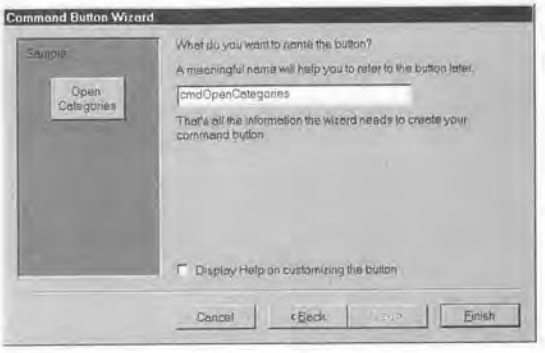


FIG. 30.12 Specifying the name of the command button.

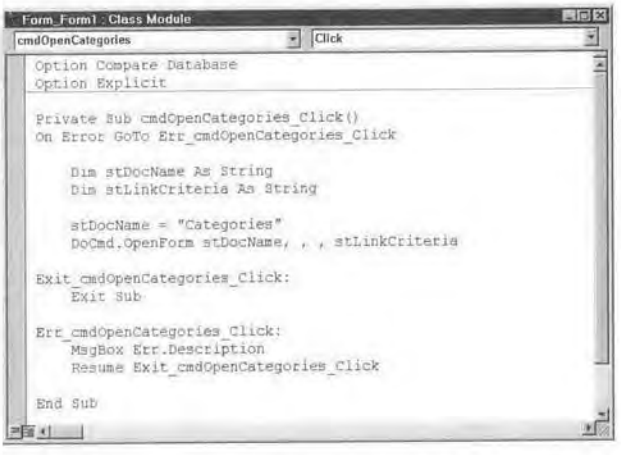


FIG. 30.13 The event-handling subprocedure for the Open Categories command button.

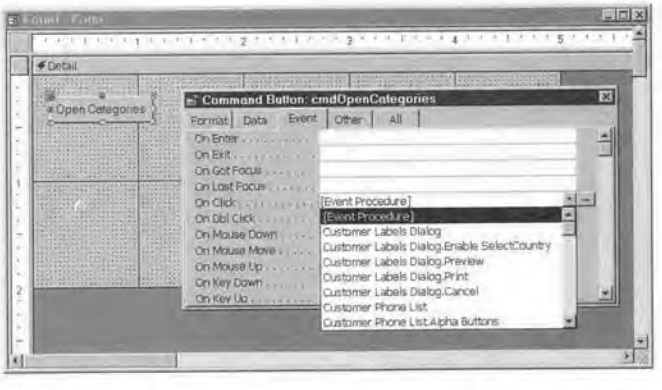


FIG. 30.14 Selecting between a subprocedure and an existing Access macro to handle an event.

Oracle Ex. 1009, p. 1068
Oracle v. DeBellis
IPR2024-00221

Using Functions to Respond to Events. You can create your own Main Switchboard form by adding additional command buttons to the form in order to open other forms. It's much more efficient, however, to use a single procedure to perform a set of identical tasks in which only the name of the form changes. Minimizing the amount of code in a form speeds opening of the form and minimizes the size of your database file.

Access lets you call a function and pass one or more parameter (argument) values to the function in response to events. A function (not a subprocedure) is required, despite the fact that Access disregards the return value, if any, of the function. Control Wizards won't write the function code for you, nor will the Code Builder create a function stub. You must write the function yourself before calling it from an event. The `OpenForms` function in preceding Listing 30.1 is an example of using a function as an event handler.

You can easily change code written by the Command Button Wizard to a general-purpose function that opens any form whose name you pass as an argument. Figure 30.15 shows a simple modification of the `cmdOpenCategories_Click` subprocedure (refer to Figure 30.13), substituting a user-defined function for the event handler. When you substitute **`Function`** for **`Sub`** in the first line, the VBA interpreter automatically changes **`Exit Sub`** to **`Exit Function`** and **`End Sub`** to **`End Function`**. You change the name of the function, add the `strFormName` parameter, pass the `strFormName` parameter to the `OpenForm` action, and eliminate code that's not needed for the function.



FIG. 30.15 A simple modification of the subprocedure in Figure 30.13 for creating an event-handling function.

Note

If you don't change the name of the subprocedure or pass the caret (insertion point) through the line containing the **`Function`** reserved word when converting from a subprocedure to a function, you receive a compile error. The VBA interpreter holds the existing subprocedure name in memory until the line is reinterpreted. Thus, creating a function of the same name results in a duplicate procedure name in the same class module if you simply press the Enter key when making the change. Duplicate procedure names aren't permitted within the same module, nor are duplicate names of **`Public`** procedures permitted within the same project.

Oracle Ex. 1009, p. 1069
Oracle v. DeBellis
IPR2024-00221

The simple changes to the subprocedure code shown in Figure 30.15 violate several good VBA programming practices. All variables and function return values should be assigned an explicit data type, and error-handling code should reflect the name of the function. Figure 30.16 shows the final version of the `cmdOpenForm` function.

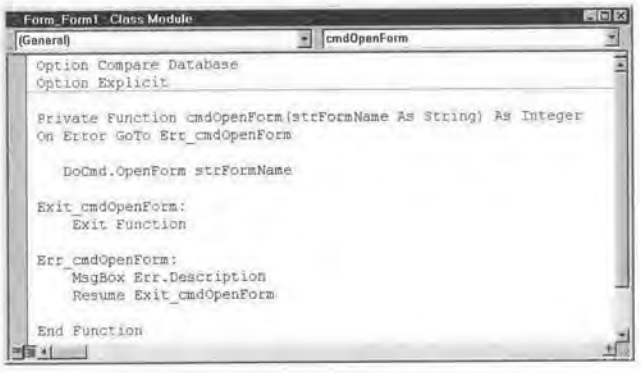


FIG. 30.16 Cleaning up the code of the `cmdOpenForm` function.

The syntax to enter in the event text box for executing a function is

```
=FunctionName([Argument1[, Argument2[, ...]])
```

The arguments are optional, but unless you pass an argument value, such as a form name, there is no advantage to using a function call as an event handler. Arguments must be passed as literal values, such as "FormName" or a numeric value. Figure 30.17 shows the entry you type in the On Click text box to open the Categories form, **=cmdOpenForm("Categories")**. To create buttons to open other forms, copy the command button to the Clipboard, paste the copy to your form, and change Categories to the name of the form you want to open.

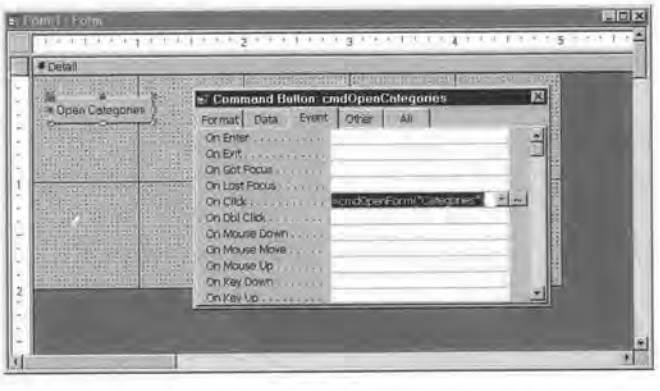


FIG. 30.17 Specifying the function name and literal argument value for an event handler.

Oracle Ex. 1009, p. 1070
Oracle v. DeBellis
IPR2024-00221

Understanding Access 97's Event Repertoire

When you interact with an object by using the keyboard or the mouse, you can change the object's state. The object's state is stored with the other data about the object. Access makes some of the changes in the object's state available as opportunities to interrupt normal processing. These special changes in an object's state are called *events*. An event is a change in the state of an object at which you can interrupt normal processing and define a response.



Not all Access objects trigger (or *fire*) events. Forms and reports were the only database objects for which Access 95 and earlier databases defined events. All forms, form sections, and controls on a form trigger events; all reports and report sections also trigger events. Access 2.0 defined 35 events; Access 95 added four more: `Filter`, `ApplyFilter`, `NoData`, and `Page`. Access 97 offers the following four new events: `Initialized`, `ItemAdded`, `ItemRemoved`, and `Terminate`. These four new events are the first to trigger from objects other than forms, reports, or control objects.

The best way to understand events is to categorize each by the type of action that causes the event to occur. There are 11 categories:

- *Mouse events* are triggered when you click form objects.
- *Keyboard events* are triggered by forms and form controls when you type or send keystrokes with the `SendKeys` action while the form object has the focus.
- *Window events* are triggered by opening or closing forms or reports.
- *Focus events* are triggered when a form or form control gains or loses the focus or when a form or report becomes active or inactive.
- *Data events* are triggered by forms and form controls when you change data in controls or records, or by forms when the focus moves from one record to another.
- *Filter events* are triggered by forms when you apply or remove filters.
- *Print events* are triggered by reports and report sections when you print or preview a report.
- *Error events* are triggered by a form or report that has the focus when an error occurs.
- *Timing events* are triggered by forms when a specified time interval passes.



- *Class module events* fire when you open or close an instance of a class.



- *Reference events* fire when you add or remove a reference to an object or type library in the new `References` collection.

Table 30.1 groups Access 97's events according to their source.

Oracle Ex. 1009, p. 1071
Oracle v. DeBellis
IPR2024-00221

Table 30.1 Events Grouped by Cause

Category	Source	Events

Mouse events	The user creating mouse actions.	`Click`
		`DblClick`
		`MouseDown`
		`MouseUp`
		`MouseMove`

Keyboard events	The user typing on the keyboard or `SendKeys` sending keystrokes.	`KeyDown`
		`KeyUp`
		`KeyPress`

Window events	Opening, closing, or resizing a window.	`Open`
		`Load`
		`Unload`
		`Close`
		`Resize`

Focus events	An object losing or gaining the focus, or a form or report becoming active or inactive.	`Enter`
		`GotFocus`
		`Exit`
		`LostFocus`
		`Activate`
		`Deactivate`

Data events	Making changes to a control's data, displaying records in a form, or moving the focus from one record to another in a form.	`Current`
		`BeforeInsert`
		`AfterInsert`
		`Delete`
		`BeforeDelConfirm`
		`AfterDelConfirm`
		`BeforeUpdate`
		`AfterUpdate`
		`Change`
		`Updated`
		`NotInList`

Filter events	Opening or closing a filter window, or applying or removing a filter.	`Filter` `ApplyFilter`

Print events	Selecting or arranging data for printing.	`Format`
		`Print`
		`Retreat`
		`NoData`
		`Page`

Error event	Generating an error.	`Error`
Timing event	A specified period of time expiring.	`Timer`
Class module events	Opening a new instance of a class module or terminating an instance of a class module.	`Initialize` `Terminate`
Reference event	Adding or removing a reference to an object or type library.	`ItemAdded` `ItemRemoved`

Each event that an object triggers has a corresponding event property listed in a separate category of the object's property sheet. Table 30.2 lists the events for each object. (References and instances of class modules aren't objects, so the new Access 97 events don't appear in Table 30.2.) In Table 30.2, control objects are listed first, followed by form and report objects (according to their order in the Toolbox).

Oracle Ex. 1009, p. 1072
Oracle v. DeBellis
IPR2024-00221

Table 30.2 Events Classified by Object Type

Object	Mouse	Keyboard	Window
Label	Click DblClick MouseDown MouseUp MouseMove		
Text Box	Click DblClick MouseDown MouseUp MouseMove	KeyDown KeyUp KeyPress AfterUpdate	Enter Exit BeforeUpdate
Option Group	Click DblClick MouseDown MouseUp MouseMove	Enter Exit	BeforeUpdate AfterUpdate
Toggle Button	Click DblClick MouseDown MouseUp MouseMove	KeyDown KeyUp KeyPress AfterUpdate	Enter Exit BeforeUpdate
Option Button	Click DblClick MouseDown MouseUp MouseMove	KeyDown KeyUp KeyPress AfterUpdate	Enter Exit BeforeUpdate
Check Box	Click DblClick MouseDown MouseUp MouseMove	KeyDown KeyUp KeyPress AfterUpdate	Enter Exit BeforeUpdate
Combo Box	Click DblClick MouseDown MouseUp MouseMove	KeyDown KeyUp KeyPress AfterUpdate	Enter Exit BeforeUpdate
List Box	Click DblClick MouseDown MouseUp MouseMove	KeyDown KeyUp KeyPress AfterUpdate	Enter Exit BeforeUpdate
Command Button	Click DblClick MouseDown MouseUp MouseMove	KeyDown KeyUp KeyPress	Enter Exit
Image	(None)		
Object Frame	Click DblClick MouseDown MouseUp MouseMove	Enter Exit	GotFocus LostFocus

Oracle Ex. 1009, p. 1073
Oracle v. DeBellis
IPR2024-00221

Focus	Data and Filter	Print	Error and Timing
GotFocus LostFocus	Change		
GotFocus LostFocus			
GotFocus LostFocus			
GotFocus LostFocus			
GotFocus LostFocus	Change NotInList		
GotFocus LostFocus			
GotFocus LostFocus			
Updated			

(continues)

Oracle Ex. 1009, p. 1074
Oracle v. DeBellis
IPR2024-00221

Table 30.2 Continued

Object	Mouse	Keyboard	Window
Bound Object Frame	Click DblClick MouseDown MouseUp MouseMove	KeyDown KeyUp KeyPress AfterUpdate	Enter Exit BeforeUpdate
Subform/ Subreport	Enter Exit		
Page Break			
Line Rectangle	Click DblClick MouseDown MouseUp MouseMove		
Tab Control	Click DblClick MouseDown MouseUp MouseMove	KeyDown KeyUp KeyPress	Enter Exit
Form Sections	Click DblClick MouseDown MouseUp MouseMove		
Form	Click DblClick MouseDown MouseUp MouseMove Activate Deactivate Current Delete	KeyDown KeyUp KeyPress Close Resize AfterInsert BeforeDelConfirm AfterDelConfirm	Open Load Unload AfterUpdate BeforeInsert
Report Page Header/Footer	Format Print		
Group Header/ Footer	Format Print Retreat		
Report Detail section	Format Print Retreat		
Report	Open Close Activate Deactivate	NoData	

Oracle Ex. 1009, p. 1075
Oracle v. DeBellis
IPR2024-00221

Focus	Data and Filter	Print	Error and Timing
GotFocus LostFocus	Updated		
GotFocus LostFocus	Change		
GotFocus LostFocus BeforeUpdate	Filter ApplyFilter	Error Timer	

Oracle Ex. 1009, p. 1076
Oracle v. DeBellis
IPR2024-00221

Usually the corresponding event property is the event name preceded by the word On; for example, the `Click` event triggered by a command button becomes the On Click property in the button's property sheet. Figure 30.18 shows the Event page of the property sheet for a text box displaying the 15 events that the text box control can trigger. Notice that all event properties—except the Before Update and After Update data event properties—follow the pattern of preceding the event name with `On`.



FIG. 30.18 The event names that are applicable to a text box control.

Working with Access 97's *DoCmd* Methods

Some of the `DoCmd` methods duplicate menu commands, such as Print, Close, and Apply Filter/Sort. Other methods substitute for mouse actions. For example, you can use the `SelectObject` method to select a database object in the same way that you select an open window by clicking it or select a database object in the Database window by clicking the object's name. Other `DoCmd` methods provide capabilities that are not available through menu commands, such as `Beep`, which emits a beep sound, or `MsgBox`, which displays a custom message.



Table 30.3 lists available `DoCmd` methods grouped by task. Access 2.0 provided 47 macro actions; Access 95 added two new `DoCmd` methods: `Save` and `SetMenuItem`. Access 97 replaces the `DoMenuItem` action or method with the `RunCommand` method, which lets you execute any native menu choice or standard toolbar button. The `AddMenu` item in the Method column of Table 30.3 is an Access 95 and earlier macro action; it is not a method of the `DoCmd` object and cannot be executed from Access VBA code.

Table 30.3 `DoCmd` Methods Grouped by Task

Category	Task	Method
Manipulating	Copy or rename a database object.	`CopyObject`, `Rename`
	Delete a database object.	`DeleteObject`
	Open a table, query, form, report, or module.	`OpenTable`, `OpenQuery`, `OpenForm`, `OpenReport`, `OpenModule`

Oracle Ex. 1009, p. 1077
Oracle v. DeBellis
IPR2024-00221

Category	Task	Method
	Close a database object.	`Close`
	Save a database object.	`Save`
	Print a database object.	`Print`, `OpenForm`, `OpenQuery`, `OpenReport`
	Select a database window object.	`SelectObject`
	Copy or rename an object.	`CopyObject`, `Rename`
	Update data or update the screen.	`RepaintObject`, `Requery`, `ShowAllRecords`
	Set the value of a field, control, or property.	`SetValue`
Executing	Carry out a menu command.	`RunCommand`
	Run a query.	`OpenQuery`, `RunSQL`
	Run a macro or a VBA procedure.	`RunMacro`, `RunCode`
	Run another Windows or DOS application.	`RunApp`
	Stop execution of a macro.	`StopMacro`, `StopAllMacros`
	Stop execution of Access.	`Quit`
	Stop execution following an event.	`CancelEvent`
Working with data in forms and reports	Select or sort records Find a record.	`ApplyFilter` `FindRecord`, `FindNext`
	Move to a particular location.	`GoToControl`, `GoToRecord`, `GoToPage`
Importing and exporting data	Output data from a table, query, form, report, or module in .xls, .rtf, or .txt formats.	`OutputAs`
	Include in an e-mail message data from a table, query, form, report, or module in .xls, .rtf, or .txt format.	`SendObject`
	Transfer data between Access and other data formats.	`TransferDatabase`, `TransferSpreadsheet`, `TransferText`
Miscellaneous	Create a custom menubar.	`AddMenu`, `SetMenuItem`
	Sound a beep.	`Beep`
	Display or hide a toolbar.	`ShowToolbar`
	Send keystrokes to Access or a Windows application.	`SendKeys`

(continues)

Oracle Ex. 1009, p. 1078
Oracle v. DeBellis
IPR2024-00221

Table 30.3 `DoCmd` Methods Grouped by Task

Category	Task	Method
	Display an hourglass.	`Hourglass`
	Display or hide system information.	`Echo`, `SetWarnings`
	Display custom messages.	`MsgBox`

Note

Methods applicable to macros—such as `RunMacro`, `RunCode`, `StopMacro`, and `StopAllMacros`—are obsolete in Access 97; these `DoCmd` methods are included for backward compatibility only.

Arguments of *DoCmd* Methods

Most `DoCmd` methods require additional information as arguments to specify how the method works. For example, when you use the `OpenForm` method, you must specify the name of the form to open as the `strFormName` argument. Also, to specify whether you want to display the Form, Design, Print Preview, or Datasheet view, you must use the `intView` argument. To specify whether you want to allow editing or adding new records, you must use the `intDataMode` argument. Finally, to specify whether you want the form to be hidden, behave like a dialog, or be in normal mode, you must use the `intWindowMode` argument. You specify the values of arguments of the **`Integer`** data type by substituting Access intrinsic constants, which use the `ac` prefix, as in

```
DoCmd.OpenForm strFormName, acNormal, strFilterName, strCriterion, acEdit,
acDialog, strOpenArg
```

The `acNormal`, `acEdit`, and `acDialog` argument values are Access intrinsic constant values for the `intView`, `intDataMode`, and `intWindowMode` arguments, respectively. You also can specify the numeric value of the constant, but there is no guarantee that the numeric values of Access constants will remain the same in future versions of Access. Thus using the names of Access intrinsic constants is a better programming practice than supplying numeric values for method arguments. Access 97's online help for `DoCmd` methods lists the Access constants that are applicable to each argument of the method.

Table 30.4 is an alphabetical list of `DoCmd` methods, together with the descriptive names of the argument(s) applicable to each method. Methods marked with an asterisk (*) are not recommended for new Access applications and are included for backward compatibility with existing macros.

Table 30.4 `DoCmd` Methods, Argument Names, and Purpose

Method	Argument Name	Purpose
AddMenu*	Menu Name Menu Macro Name	Adds a drop-down menu choice to a custom menubar or adds a custom shortcut menu choice.
	Status Bar Text	Replaced by programmable CommandBars in Access 97.

Oracle Ex. 1009, p. 1079
Oracle v. DeBellis
IPR2024-00221

Method	Argument Name	Purpose
`ApplyFilter`	Filter Name Where Condition	Filters the data available to a form or report using a filter, query, or SQL `WHERE` clause.
`Beep`	No arguments	Produces a beep tone for use in warnings or alerts.
`CancelEvent`	No arguments	Cancels the normal processing that follows the event. This action is useful if a user enters invalid data in a record; then the macro can cancel the update of the database. See Help on `CancelEvent` for a list of applicable events.
`Close`	Object Type	Closes the active (default) window or a specified Object Name window.
`CopyObject`	Destination Database New Name	Duplicates the specified database object in another database or the original by using a different name.
`Delete Object`	Object Type Object Name	Deletes the Specified object. Leaves the arguments blank to delete the object selected in the Database window.
`DoMenuItem`	N/A	Replaced by `RunCommand`.
`Echo`	Echo On Status Bar Text	Turns screen refresh on or off during code execution. Hides results until they are complete and speeds code operation.
`FindNext`	No arguments	Finds the next record specified by the `FindRecord` or the `Find` method.
`FindRecord`	Find What Match Case Direction Search As Formatted Search In Find First	Finds the next record after the current record that meets the specified Criteria. Searches through a `Table`, `Form`, or `Recordset` object.
`GoToControl`	Control Name	Moves the focus to the specified field or control in the current record of the open form, form datasheet, table datasheet, or query datasheet. To move the focus to a subform's control, use the `GoToControl` method twice; first to move to the subform control and then to move to the control on the subform.
`GoToPage`	Page Number Right Down by Tab	Selects the first field in the tab order on the designated page in a multipage form.

(continues)

Oracle Ex. 1009, p. 1080
Oracle v. DeBellis
IPR2024-00221

Table 30.4 Continued

Method	Argument Name	Purpose
`GoToRecord`	Object Type Object Name Record Offset	Displays the specified record in an open table, form, or query datasheet.
`Hourglass`	Hourglass On	Displays an hourglass in place of the mouse pointer during execution. Use this action while running long procedures.
`Maximize`	No arguments	Maximizes the active window.
`Minimize`	No arguments	Minimizes the active window to an icon within the Access window.
`MoveSize`	Right Down Width Height	Moves or changes the size of the active window.
`MsgBox*`	Message Beep Type Title	Displays a warning or informational message box and waits for the user to click the OK button. Replaced by VBA **`MsgBox`** command.
`OpenForm`	Form Name View Filter Name Where Condition Data Mode Window Mode	Opens or activates a form in one of its views. You can restrict the form to data-matching criteria, different modes of editing, and whether the form acts as a modal or popup dialog.
`OpenModule`	Module Name Procedure Name	Opens the specified module and displays the specified procedure.
`OpenQuery`	Query Name View Data Mode	Opens a select or crosstab query or runs an action query.
`OpenReport`	Report Name View Filter Name Where Condition	Opens a report in the view that you specify and filters the records before printing.
`OpenTable`	Table Name View Data Mode	Opens or activates a table in the view that you specify. You can specify the Data-Entry or Edit mode for tables in Datasheet view.
`OutputTo`	Object Type Object Name Output Format Output File Autostart	Copies the data in the specified object to a Microsoft Excel (.xls), rich-text format (.rtf), or to a DOS text (.txt) file. `Autostart = Yes` starts the application with the association to the extension.

Oracle Ex. 1009, p. 1081
Oracle v. DeBellis
IPR2024-00221

Method	Argument Name	Purpose
Print	Print Range Page From Page To Print Quality Copies Collate Copies	Prints the active datasheet, report, or form.
Quit	Options	Closes Access, saving altered objects according to the command that you specify.
Rename	New Name	Renames the object selected in the Database window.
RepaintObject	Object Type Object Name	Forces pending recalculations and screen updates for the controls of the specified database object or the active database object if you leave the arguments blank. Does not show new, changed, or deleted records from the object's underlying source.
Requery	Control Name	Updates the data in the specified control by repeating the control's query if the control is based on a query or by displaying new, changed, or deleted records if the control is based on a table. Leave the argument blank to requery the source of the active object.
Restore	No arguments	Restores a maximized or minimized window to its previous window.
RunApp*	Command Line	Runs a Windows- or an MS-DOS-based application. Access 97 uses the VBA Shell function to run other applications.
RunCode*	Function Name	Runs a user-defined function written in VBA. Replaced by direct execution of code.
RunCommand	Menu Bar Menu Name Command Subcommand	Runs any command on a built-in Access 97 menubar or toolbar if the bar is appropriate for the view when the macro carries out the command.
RunMacro*	Macro Name Repeat Count Repeat Expression	Runs the specified macro. Enter the macro name's full syntax to run an individual macro in a macro group. Use the Repeat Count and Repeat Expression arguments to specify how many times to run the macro. Replace macros with VBA procedures in Access 97 applications.

(continues)

Oracle Ex. 1009, p. 1082
Oracle v. DeBellis
IPR2024-00221

Table 30.4 Continued

Method	Argument Name	Purpose
RunSQL*	SQL Statement	Runs an action query as specified by the SQL statement. (To run a select query, use the OpenQuery action instead.) Replaced by the Execute method of a QueryDef object.
Save	Object Type Object Name	Saves the specified database object. Leave the arguments blank to save the active window.
SelectObject	Object Type Object Name In Database Window	Selects a specified database object.
SendKeys	Keystrokes Wait	Sends keystrokes to any active Windows application.
SendObject	Object Type Object Name Output Format To Cc Bcc Subject Message Text Edit Message	Sends the specified datasheet, form, report, or module in an electronic mail message. You can't send a macro. This action requires that you have a MAPI-compliant electronic mail application on your computer.
SetMenuItem	Menu Index Menu Item Menu Sub Item Flag	Sets the state of a menu item on a custom menu to check a command or make it unavailable (grayed, or disabled).
SetValue*	Item Expression	Sets the value of a field, control, or property on a form, form datasheet, or report. Replaced by VBA code to set values.
SetWarnings	Warnings On	Turns default warning messages on or off. Does not suppress error messages or system dialogs that require you to input text or select an option.
ShowAllRecords	No arguments	Removes any filters and requeries the active object.
StopAllMacros*	No arguments	Stops all macros that are currently running. VBA code replaces macros in Access 97.
StopMacro*	No arguments	Stops the current macro. VBA code replaces macros in Access 97.

Oracle Ex. 1009, p. 1083
Oracle v. DeBellis
IPR2024-00221

Method	Argument Name	Purpose
`TransferDatabase`	Transfer Type Database Type Database Name Object Type Source Destination Structure Only	Imports data from another database, exports data to another database, or links a table in another database to the current database. The other database can be an Access or SQL database.
`TransferSpreadsheet`	Transfer Type Spreadsheet Type Table Name File Name Has Field Names Range	Imports data from a spreadsheet file or exports Access data to a spreadsheet file.
`TransferText`	Transfer Type Specification Name Table Name File Name Has Field Names	Imports data from a text file or exports Access data to a text file.

The `OutputTo`, `TransferDatabase`, `TransferSpreadsheet`, and `TransferText` methods deserve special attention by application developers. These bulk transfer methods greatly simplify the data interchange between Access and other Office 97 applications, such as Excel and Word. The "Converting Access Macros to VBA Code" section later in this chapter illustrates the power of the `TransferText` method. The more complex `DoCmd` methods, together with Access 97's flexible report generation capabilities, are often the deciding factor when choosing between Visual Basic 4+ or Access 97 for developing database front ends. Visual Basic does not offer equivalents of the bulk transfer Access methods.

Replacing *AddMenu* Macro Actions with *CommandBar* Objects



Prior versions of Access required the use of multiple menu macros with the `AddMenu` macro action to create custom menu bars. Access 95 included a Menu Builder Add-In, similar to that of Visual Basic, to make creating custom menus easier. All Office 97 applications replace conventional menu bars and toolbars with `CommandBar` objects. `CommandBar` objects create menubars, popup menus, toolbar buttons, and combo boxes. To gain VBA programming access to `CommandBar` objects, your database must include a reference to the Microsoft Office 8.0 Object Library, Mso97.dll (see Figure 30.19). Mso97.dll includes other objects that are common to Office 97, such as the `Balloon` object of the Office `Assistant` object and the `DocumentProperty` object for files. Figure 30.20 shows the Microsoft Office Objects help topic that displays the collections and objects exposed by Mso97.dll. The top-level collections and objects—`CommandBars`, `Assistant`, `FileSearch`, and `DocumentProperties`—are members of Access 97's `Application` object.

Oracle Ex. 1009, p. 1084
Oracle v. DeBellis
IPR2024-00221

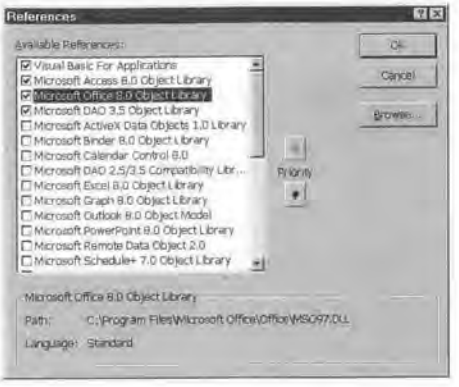


FIG. 30.19 Creating a reference to the Microsoft Office 8.0 Object Library.

With a reference to Mso97.dll, Office 97 object classes appear in the Object Browser's window when you select the Office library (see Figure 30.21). Selecting a class or member of the class and then clicking the Object Browser's Help (?) button displays the Microsoft Office Visual Basic help topic for the selection. Using Object Browser's Help button is the easiest way to obtain the VBA syntax for programming specific Office objects.



FIG. 30.20 The hierarchy of Office 97 collections and objects displayed by the Microsoft Office Objects help topic.

Oracle Ex. 1009, p. 1085
Oracle v. DeBellis
IPR2024-00221

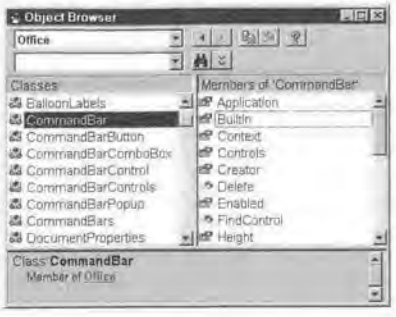


FIG. 30.21 A few of the Office object classes and members displayed in Object Browser.

The Debug window is useful for gaining familiarity with programming Office objects. Figure 30.22 shows how to obtain property values of the `CommandBars` collection, CommandBar objects, and the `Control` objects of `CommandBars`. Unlike most other Access collections that begin with an index value of 0, the first member of Office collections has an index value of 1. The `Application.` preface in the first statement of Figure 30.22 is optional; the `Application` object is assumed when referring to top-level Access objects. To view the members of the Access `Application` class, open the Access library in Object Browser and select Application in the Classes list.

Note

To open the Debug window, open a form in Design view, and click the Code button of the toolbar to display the class module for the form, then choose View, Debug Window or press Ctrl+G.

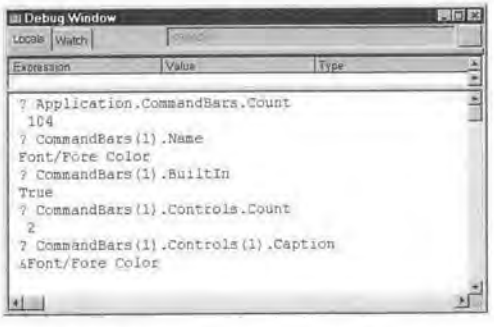


FIG. 30.22 Using the Debug window to obtain property values of CommandBar and Control.

The following simple VBA subprocedure iterates the `CommandBars` collection and prints the `Name` and `Visible` property values of each CommandBar in the Debug window:

```
Dim msoBar As CommandBar

Sub PrintCommandBars()
   For Each msoBar In CommandBars
      Debug.Print msoBar.Name, msoBar.Visible
   Next msoBar
End Sub
```

Oracle Ex. 1009, p. 1086
Oracle v. DeBellis
IPR2024-00221

Northwind.mdb has a custom command bar that adds the Show Me entry to Access's main menubar. When you run the preceding subprocedure from the Debug window, the last entry in the list is `NorthwindCustomMenuBar` with its `Visible` property **`True`**. Figure 30.23 shows some of the property values of `NorthwindCustomMenuBar`. The Show Me menu choice is the 15th member of the Controls collection of the menubar, with a `Caption` property value of `"Show Me"`. The `OnAction` property value, `"=ShowHelpAPI()"`, uses the `ShowHelpAPI` function of the Northwind Help Module to open the help file for the Northwind Traders sample database. To display the `ShowHelpAPI` function, open a module, choose Edit, Find, then search for `ShowHelp` with the Current Database option selected.

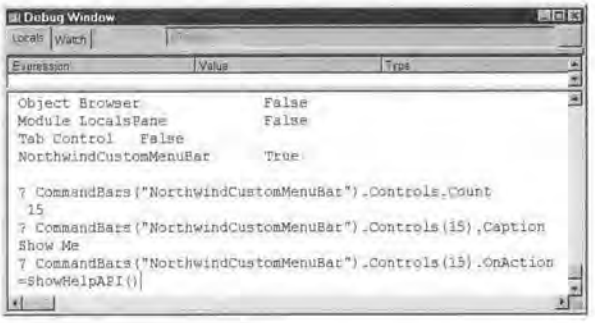


FIG. 30.23 Property values of the `NorthwindCustomMenuBar` and its added `Control` object, Show Me.

You can change the properties of existing command buttons with VBA code. As an example, typing the following code in the Debug window immediately changes the Show Me menu choice to Show You:

```
CommandBars("NorthwindCustomMenuBar").Controls(15).Caption = "Show &You"
```

You add a new `CommandBar` object by applying the `Add` method to the `CommandBars` collection, setting the properties of the new `CommandBar` object, and adding members to the `CommandBar` object's `Controls` collection. A full exposition of VBA programming of custom `CommandBar` objects is beyond the scope of this book. Most of the developer-level books for Access 97 listed in the "Bibliography" section of the Introduction to this book cover custom `CommandBar` programming in detail.

Note

The Command Bar Browser (CommandBarsForm) form of the Developer Solutions database (Solutions.mdb) in the \Program Files\Microsoft Office\Office folder provides a range of VBA programming examples for the `CommandBar` object. Solutions.mdb isn't installed when you choose the Typical setup option. If you didn't install Solutions.mdb, run Setup.exe from the Office 97 CD-ROM, click Add/Remove, and then specify installation of the Developer Solutions database in the Sample Databases category of the Access Setup options.

Oracle Ex. 1009, p. 1087
Oracle v. DeBellis
IPR2024-00221

Replacing the AutoExec Macro and Specifying a Custom *CommandBar* with Startup Properties

Access 2.0 and earlier databases required you to write an AutoExec macro to display a specified form automatically upon opening a database with Access. Commencing with Access 95, you replace AutoExec macros with database properties that you specify in the Startup dialog. In addition to a specified opening form, you also can assign an application title to the database and specify the name of a custom `CommandBar` as Access's main menu bar. To set startup properties, follow these steps:

1. Make the Database window active and choose Tools, Startup to open the Startup dialog. Click the Advanced button to expand the dialog.
2. Type a name for your application in the Application Title text box; the application name replaces Access's default title bar caption. If you have a special icon file (*Filename*.ico), you can specify the icon file in the Application Icon text box.
3. If you've created a custom `CommandBar` for your application, select the `CommandBar` in the drop-down Menu Bar list.
4. Open the Display Form list and select the form that you want to appear when you open the database (see Figure 30.24). You can elect to hide the Database window and the status bar by clearing the Display Database Window and Display Status Bar check boxes.

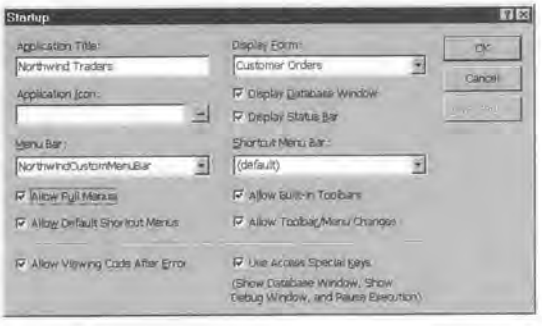


FIG. 30.24 Setting the Application Title, Menu Bar, and Display Form properties in the Startup dialog.

5. The remaining check boxes let you limit use of your application's menu bars, shortcut menus, and toolbars. You also can restrict viewing of VBA code when an error occurs and disable Access's special key combinations.

Tip

Don't clear any of the Startup dialog's check boxes until you're ready to release your application to users. Limiting Access's built-in menu, toolbar, and code-viewing feature set is likely to impede your development activities.

6. Click OK to assign the changes you made and close the Startup dialog.

Oracle Ex. 1009, p. 1088
Oracle v. DeBellis
IPR2024-00221

Converting Access Macros to VBA Code

Many existing Access applications make extensive use of macros to automate form and report operations. You can use Access 97's macro-to-VBA code conversion feature to change all but AutoExec and menu macros to Access VBA procedures. Converting macros to VBA code aids in learning VBA programming in much the same manner as recording Excel 97 or Word 97 macros and then examining the resulting VBA code. The Wildfeeds application (30_wild.mdb), used in this conversion example, began as an Access 2.0 demonstration of the power of Access macros. Wildfeeds imports a fixed-width text file, Wildfeed.txt, that lists unscheduled satellite video programming available to C-band dish owners. Wildfeeds also demonstrates the power of Access's `TransferText` method.

Note

Wildfeeds is published monthly by Gary Bourgois as "Birdfeed" and is available in text format from **http://www.nmia.com/~roberts/wildfeed** (save the HTML page as Wildfeed.txt). The older version of Wildfeed.txt used with the Wildfeed application described in this section uses 12-hour HH:MM A/P starting time designations. The current version of "Birdfeed" uses 24-hour starting times (HHMM). You might find it an interesting exercise to modify 30_wild.mdb to accommodate the new text format with the 24-hour time field. (The 24-hour time format simplifies the programming considerably.)

The columns of Wildfeed.txt—shown opened in Notepad in Figure 30.25—list the name of the program, the days of the week that the program appears (MF = Monday through Friday), the program starting time (EST), the program's satellite abbreviation (G4 = Galaxy 4) and transponder number, and the classification of the program. 30_wild.mdb includes an import specification that parses the lines of text into designated fields of the tblWildFeed table. Figure 30.26 shows the imported text file opened in the sbfWildFeed subform. Clicking the Yesterday, Today, or Tomorrow button redisplays the subform according to the day of the week.

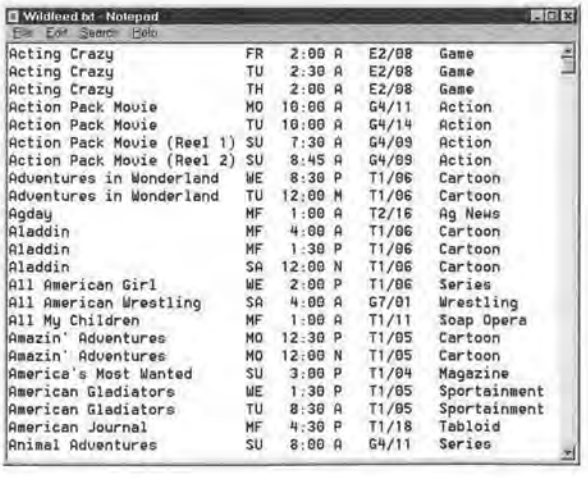


FIG. 30.25 An early version of the Wildfeeds.txt file opened in Windows Notepad.

Oracle Ex. 1009, p. 1089
Oracle v. DeBellis
IPR2024-00221

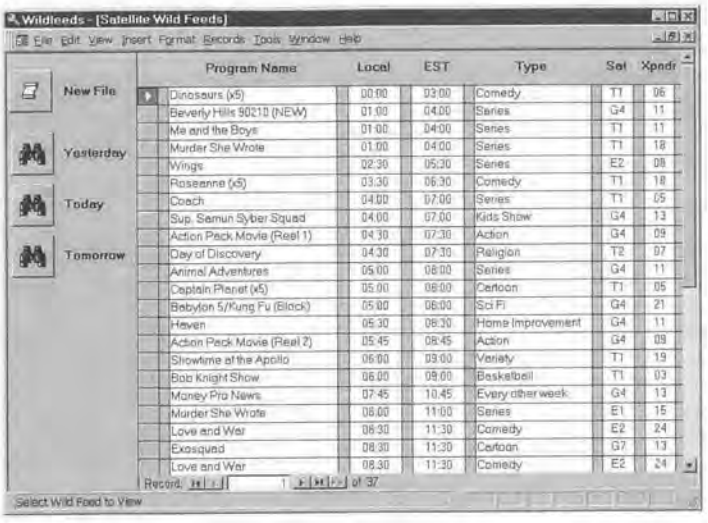


FIG. 30.26 The main form and subform of the Wildfeeds application displaying satellite programs sorted by local time (PST or PDT).

The original mcrWildFeed macro, executed by clicking the New File button, consists of a chain of 17 macro actions—13 of which are shown in the mcrWildFeeds macrosheet in Figure 30.27. Converting the Access 95 version of 30_wild.mdb into an Access 97 database changes `DoMenuItem` actions to `RunCommand` actions. (`DoCmd` methods are called actions when executed from macrosheets.) The selected `TransferText` action imports the Wildfeed.txt file in accordance with the WildFeed import specification. Four queries are required to reformat the time and day-of-the week values for correct sorting. The remainder of the macro actions reopen frmWildFeed and sbfWildFeed to display the newly imported data.

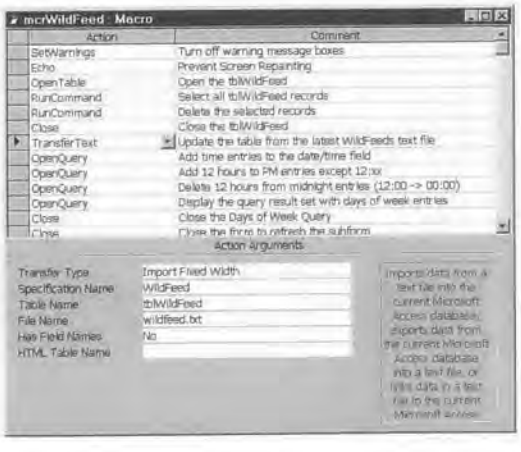


FIG. 30.27 The mcrWildFeed macrosheet after converting 30_wild.mdb to an Access 97 database.

Oracle Ex. 1009, p. 1090
Oracle v. DeBellis
IPR2024-00221

Note

The Access 95 version of 30_wild.mdb is included in the Ua97mdb.zip file available for downloading from **http://www.mcp.com/320336271804569/softlib/database/**. Ua97mdb.zip includes all of the example databases of this book.

You must have the Advanced Wizards installed to convert macros to VBA code. If you used the Typical installation for Office 97 or Access 97, the Advanced Wizards aren't installed. In this case, you must rerun Setup from the distribution CD-ROM or a network installation share. Click the Add/Remove button and specify the Advanced Wizards in the Wizards section of the Access component of Setup.

To convert Access macros in an earlier version database to Access 97 VBA code, follow these steps that use 30_wild.mdb as an example:

1. Open the database in Access 97 and select the Convert Database option. Save the database in the same directory with a new file name, such as **97_wild.mdb.** (Unlike compacting, you can't replace the original version with a converted database.)



2. Open the form whose macros you want to convert in Design view. This action lets the wizard place the generated code for the macros in the class module of the form.
3. Choose Tools, Macro, Convert Form's Macros to Visual Basic to start the conversion process.
4. When the Convert Form Macros: *FormName* dialog appears, accept the default Add Error Handling to Generated Functions and Include Macro Comments options (see Figure 30.28). Click Convert to complete the conversion operation, and then click OK when the Conversion Finished message box appears.



FIG. 30.28 Specifying inclusion of standard error-handling code and comments in the code generated by the macro-to-VBA conversion process.



5. Click the Code button of the toolbar to display the class module for the form with the added VBA code. Figure 30.29 shows part of the code generated for the mcrWildFeed macro. Note that converted class modules don't include the standard `Option Explicit` statement in the Declarations section.

Oracle Ex. 1009, p. 1091
Oracle v. DeBellis
IPR2024-00221

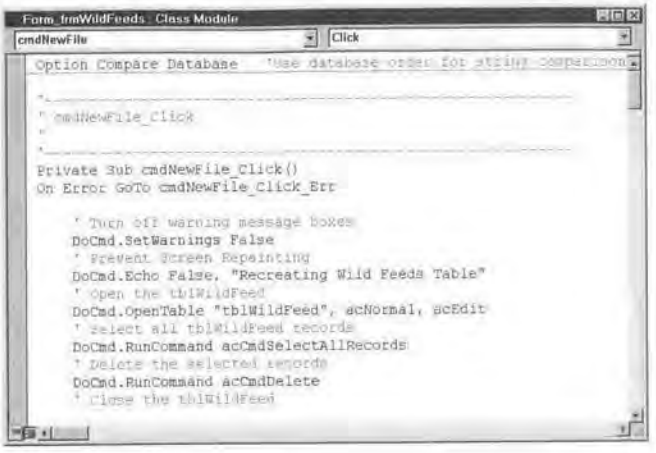


FIG. 30.29 VBA code generated for the first five macro actions of the mcrWildFeed macrosheet.

Listing 30.2 contains the event-handling code for each of the converted macros run by the On Click event of the four command buttons: cmdNewFile, cmdYesterday, cmdToday, and cmdTomorrow.

Listing 30.2 Event-Handling Code for the Four Command Buttons of `frmWildFeed` Created by the Macro Conversion Process

```
Option Compare Database   'Use database order for string comparisons

'---------------------------------------------------------
' cmdNewFile_Click
'
'---------------------------------------------------------
Private Sub cmdNewFile_Click()
On Error GoTo cmdNewFile_Click_Err

    ' Turn off warning message boxes
    DoCmd.SetWarnings False
    ' Prevent Screen Repainting
    DoCmd.Echo False, "Recreating Wild Feeds Table"
    ' Open the tblWildFeed
    DoCmd.OpenTable "tblWildFeed", acNormal, acEdit
    ' Select all tblWildFeed records
    DoCmd.RunCommand acCmdSelectAllRecords
    ' Delete the selected records
    DoCmd.RunCommand acCmdDelete
    ' Close the tblWildFeed
    DoCmd.Close acTable, "tblWildFeed"
    ' Update the table from the latest WildFeeds text file
    DoCmd.TransferText acImportFixed, "WildFeed", "tblWildFeed", _
       "wildfeed.txt", False, ""
    ' Add time entries to the date/time field
    DoCmd.OpenQuery "qryConvertTime", acNormal, acEdit
    ' Add 12 hours to PM entries except 12:xx
```

(continues)

Oracle Ex. 1009, p. 1092
Oracle v. DeBellis
IPR2024-00221

Listing 30.2 Continued

```
    DoCmd.OpenQuery "qryFixTimePM", acNormal, acEdit
    ' Delete 12 hours from midnight entries (12:00 -> 00:00)
    DoCmd.OpenQuery "qryFixMidnight", acNormal, acEdit
    ' Display the query result set with days of week entries
    DoCmd.OpenQuery "qryDaysOfWeek", acNormal, acEdit
    ' Close the Days of Week Query
    DoCmd.Close acQuery, "qryDaysOfWeek"
    ' Close the form to refresh the subform
    DoCmd.Close acForm, "frmWildFeeds"
    ' Open the form
    DoCmd.OpenForm "frmWildFeeds", acNormal, "", "", acEdit, acNormal
    ' Maximize the form
    DoCmd.Maximize
    ' Turn screen painting on
    DoCmd.Echo True, "Select Wild Feed"
    ' Turn warnings back on
    DoCmd.SetWarnings True

cmdNewFile_Click_Exit:
    Exit Sub

cmdNewFile_Click_Err:
    MsgBox Error$
    Resume cmdNewFile_Click_Exit

End Sub

'---------------------------------------------------------------
' cmdYesterday_Click
'
'---------------------------------------------------------------
Private Sub cmdYesterday_Click()
On Error GoTo cmdYesterday_Click_Err

    ' Turn off warnings
    DoCmd.SetWarnings False
    ' Turn off screen painting
    DoCmd.Echo False, ""
    ' Close the main and subform
    DoCmd.Close acForm, "frmWildFeeds"
    ' Open the subform
    DoCmd.OpenForm "sbfWildFeeds", acNormal, "", "", acEdit, acNormal
    ' Choose Design View
    DoCmd.RunCommand acCmdDesignView
    ' Set the RecordSource property
    Forms!sbfWildFeeds.RecordSource = "qryYesterdaysWildFeeds"
    ' Close the subform
    DoCmd.Close acForm, "sbfWildFeeds"
    ' Refresh the subform
    DoCmd.OpenForm "frmWildFeeds", acNormal, "", "", acEdit, acNormal
    ' Turn on screen painting
    DoCmd.Echo True, ""
    ' Turn on warnings
    DoCmd.SetWarnings True
```


Oracle Ex. 1009, p. 1093
Oracle v. DeBellis
IPR2024-00221

```
cmdYesterday_Click_Exit:
    Exit Sub

cmdYesterday_Click_Err:
    MsgBox Error$
    Resume cmdYesterday_Click_Exit

End Sub

'---------------------------------------------------------------
' cmdToday_Click
'
'---------------------------------------------------------------
Private Sub cmdToday_Click()
On Error GoTo cmdToday_Click_Err

    ' Turn off warnings
    DoCmd.SetWarnings False
    ' Turn off screen painting
    DoCmd.Echo False, ""
    ' Close the main and subform
    DoCmd.Close acForm, "frmWildFeeds"
    ' Open the subform
    DoCmd.OpenForm "sbfWildFeeds", acNormal, "", "", acEdit, acNormal
    ' Choose Design View
    DoCmd.RunCommand acCmdDesignView
    ' Set the RecordSource property
    Forms!sbfWildFeeds.RecordSource = "qryTodaysWildFeeds"
    ' Close the subform
    DoCmd.Close acForm, "sbfWildFeeds"
    ' Refresh the subform
    DoCmd.OpenForm "frmWildFeeds", acNormal, "", "", acEdit, acNormal
    ' Turn on screen painting
    DoCmd.Echo True, ""
    ' Turn on warnings
    DoCmd.SetWarnings True

cmdToday_Click_Exit:
    Exit Sub

cmdToday_Click_Err:
    MsgBox Error$
    Resume cmdToday_Click_Exit

End Sub

'---------------------------------------------------------------
' cmdTomorrow_Click
'
'---------------------------------------------------------------
Private Sub cmdTomorrow_Click()
On Error GoTo cmdTomorrow_Click_Err

    ' Turn off warnings
    DoCmd.SetWarnings False
```

(continues)

Oracle Ex. 1009, p. 1094
Oracle v. DeBellis
IPR2024-00221

Listing 30.2 Continued

```
        ' Turn off screen painting
        DoCmd.Echo False, ""
        ' Close the main and subform
        DoCmd.Close acForm, "frmWildFeeds"
        ' Open the subform
        DoCmd.OpenForm "sbfWildFeeds", acNormal, "", "", acEdit, acNormal
        ' Choose Design View
        DoCmd.RunCommand acCmdDesignView
        ' Set the RecordSource property
        Forms!sbfWildFeeds.RecordSource = "qryTomorrowsWildFeeds"
        ' Close the subform
        DoCmd.Close acForm, "sbfWildFeeds"
        ' Refresh the subform
        DoCmd.OpenForm "frmWildFeeds", acNormal, "", "", acEdit, acNormal
        ' Turn on screen painting
        DoCmd.Echo True, ""
        ' Turn on warnings
        DoCmd.SetWarnings True

    cmdTomorrow_Click_Exit:
        Exit Sub

    cmdTomorrow_Click_Err:
        MsgBox Error$
        Resume cmdTomorrow_Click_Exit

    End Sub
```

Inspecting the code for the three subprocedures that handle the On Click event of cmdToday, cmdYesterday, and cmdTomorrow shows that the only difference between the three procedures is the query name in the `Forms!sbfWildFeeds.RecordSource = "qry`*`QueryName`*`"` line. As discussed in the "Using Functions to Respond to Events" section earlier in the chapter, you can combine the last three subprocedures in Listing 30.2 into a single event-handling function with a parameter to accept the query name. Following are the changes you make to the code of one of the procedures:

```
Private Function cmdDateQuery (strQueryName As String) As Integer
    ...
    Forms!sbfWildFeeds.RecordSource = strQueryName
    ...
End Function
```

The VBA interpreter changes each occurrence of **Sub** to **Function** for you. It's a good programming practice to change the error-handling code to reflect the new name of the function. Once you've tested your code by choosing Debug, Compile Loaded Modules, substitute **=cmdDateQuery("qry*QueryName*")** for [Event Procedure] in the On Click text box for each of the three buttons, where **qry*QueryName*** is **qryYesterdaysWildFeeds**, **qryTodaysWildFeeds**, and **qryTomorrowsWildFeeds** for the cmdYesterday, cmdToday, and cmdTomorrow buttons, respectively.

Oracle Ex. 1009, p. 1095
Oracle v. DeBellis
IPR2024-00221

Referring to Access Objects with VBA

One of the reasons for using the term *Access VBA* in this book is that Access defines its own set of objects and uses specialized VBA syntax to refer to many Access objects. Although `Form` objects are common to most Office 97 members as well as Visual Basic, a subform (a form embedded in a form) is unique to Access. You find Report objects and subreports only in Access. The syntax for referring to a subform or subreport and for referring to controls contained in a subform or subreport is unique to Access. Even if you're an experienced Visual Basic programmer, you must become acquainted with the new syntax to write VBA code and refer to objects that are unique to Access.

Referring to Open Forms or Reports and Their Properties

You can refer to a form or report only if it is open. Access uses the `Forms` and `Reports` collections to keep track of which forms and reports are open. The `Forms` collection is the set of open forms, and the `Reports` collection is the set of open reports. Because Access lets you use the same name for a form and a report, you must distinguish between the two by specifying the collection. The syntax for the reference is the collection name followed by the *exclamation point operator* (`!`), more commonly called the *bang operator*, and the name of the form or report:

```
Forms![FormName]
Reports![ReportName]
```

Use the bang operator (`!`) to separate the collection name from the name of an object in the collection. You only need to use the square brackets (`[...]`) to enclose object names that include spaces or other punctuation that are illegal in VBA statements.

A `Form` or `Report` object has properties that define its characteristics. The general syntax for referring to a property is the object name followed by the dot operator and the name of the property:

```
Forms![FormName].PropertyName
Reports![ReportName].PropertyName
```

Use the dot (`.`) operator to separate the object's name from the name of one of its properties. For example, `Forms!frmProducts.RecordSource` refers to the `RecordSource` property of the open `frmProducts` form. You can get or set the value of the `RecordSource` property with the following two VBA statements:

```
strSource = Forms!frmProducts.RecordSource
Forms!frmProducts.RecordSource = strSource
```

To get or set the value of a property of the form in the form's own class module, you use the **`Me`** self-identifier, as in:

```
strSource = Me.RecordSource
Me.RecordSource = strSource
```

Oracle Ex. 1009, p. 1096
Oracle v. DeBellis
IPR2024-00221

The `Me` self-reference is only valid for the instance of the form open in Form view. Thus, you cannot use the two preceding statements in the Debug window unless you create a breakpoint in your code, open the form in Form view, and then execute the procedure that contains the breakpoint. Figure 30.30 shows the Debug window opened by a breakpoint and set at the first active line of code of the `ReviewProducts_Click` subprocedure of Northwind.mdb's Suppliers form. In breakpoint mode, typing **? Me.RecordSource** returns Suppliers—the name of the table to which the Suppliers form is bound.

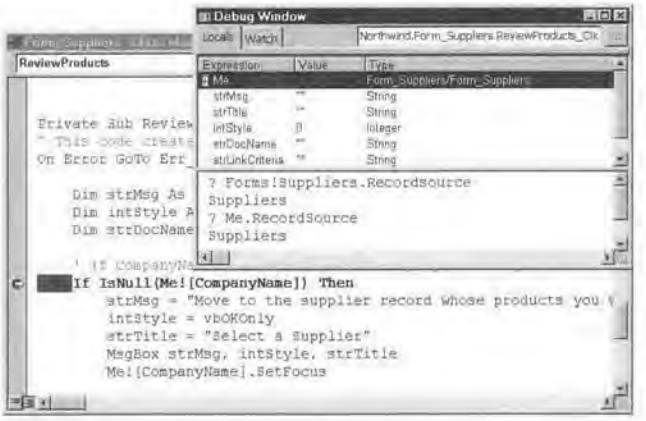


FIG. 30.30 The Debug window opened by reaching a breakpoint during VBA code execution.

◀◀ See "Adding a Breakpoint to the `IsLoaded()` Function," p. 1015

A form's property sheet lists the form properties that you can set in Design view. Forms also have properties that you can't set in Design view and that do not appear in the property sheet, such as the default `Form` property. The `Form` property refers to the collection of controls on a form. Similarly, a report's default `Report` property refers to the collection of controls in a report.

Referring to Controls and Their Properties

The following is the general syntax for referring to a control on a form or report:

```
Forms![FormName].Form![ControlName]

Reports![ReportName].Report![ControlName]
```

As before, the bang operator separates the collection name from the object name. The `Form` property is the default property that Access assumes for a form; therefore, you need not include the `Form` property explicitly in the reference. The following expression is the full identifier syntax for a form control:

```
Forms![FormName]![ControlName]
```


Oracle Ex. 1009, p. 1097
Oracle v. DeBellis
IPR2024-00221

Similarly, the following is the full identifier syntax for a report control:

```
Reports![ReportName]![ControlName]
```

For example, `Forms!frmProducts!ProductName` refers to the `ProductName` control on the open `frmProducts` form.

The syntax for referring to a control's property includes the reference to the control, followed by the dot operator, and then followed by the property name:

```
Forms![FormName]![ControlName].[PropertyName]
```

```
Reports![ReportName]![ControlName].[PropertyName]
```

For example, `Forms!frmProducts!ProductName.Visible` refers to the *`ProductName`* control's `Visible` property.

A control also has a default property. The default property of a text box is the `Text` property. To refer to the value in the `ProductName` text box control in the last example, you could use any of the following equivalent references:

```
Forms!frmProducts.Form!ProductName.Text
```

```
Forms!frmProducts!ProductName.Text
```

```
Forms!frmProducts.Form!ProductName
```

```
Forms!frmProducts!ProductName
```

Notice that the last two expressions refer both to the control's text value and to the control itself.

When you refer to a control on the active form or report, you can use a shorter version of the reference and refer to the control as follows:

```
[ControlName]
```

Likewise, you can refer to the control property as follows:

```
[ControlName].PropertyName
```

Normally, you can use either the short or full syntax to refer to a control on the active form or report. However, in some cases you must use the short syntax. For example, the `GoToControl` action's *`ControlName`* argument requires the short syntax. You can explicitly refer to a control on the form of the class module with `Me!ControlName` statements. When you refer to a control on a form or report that is not the active object, you usually must use the full identifier syntax.

Referring to Controls on a Subform or the Main Form

The key to understanding the syntax for referring to a control on a subform is to realize that the subform is a form that is bound to a subform control on the main form. The subform control has the usual attribute properties that control its display behavior, such as size and visibility, as well as linking properties that relate the records in the subform to records in the form, including the `SourceObject`, `LinkChildFields`, and `LinkMasterFields`

Oracle Ex. 1009, p. 1098
Oracle v. DeBellis
IPR2024-00221

properties. In addition, the subform control has the `Form` property. A subform control's `Form` property refers to the controls contained on the subform.

The following is the syntax for referring to the subform control:

```
Forms![FormName]![SubformControlName]
```

The syntax for referring to a control on a subform bound to a subform control is as follows:

```
Forms![FormName]![SubformControlName].Form![ControlName]
```

When the form is active, the following short syntax refers to a control on a subform of the active form:

```
[SubformControlName].Form![ControlName]
```

The `Form` property is not the subform control's default property, so you must include it explicitly in the reference. Normally, you use the subform's name as the name of the subform control. For example, if `sbfSuppliers` is the name of a form bound to a subform control also named `sbfSuppliers` on the `frmProducts` form, the following is the full syntax for referring to the `SupplierName` control on the subform:

```
Forms!frmProducts!sbfSuppliers.Form!SupplierName
```

The short syntax is as follows:

```
sbfSuppliers.Form!SupplierName
```

When the focus is in a subform's control, you can refer to a control on the main form by using the control's `Parent` property. The `Parent` property refers to the collection of controls on the main form. In the previous example, to refer to the `ProductName` control on the main form from VBA code in the class module of a subform, use the following syntax:

```
Parent!ProductName
```

All the preceding syntax examples in this section apply to reports and subreports; just change `Forms` to `Reports` and `Form` to `Report`.

> **Note**
>
> The `ShowSales_Click` subprocedure of the ShowSales form in the Developer Solutions database (Solutions.mdb) provides examples of VBA code that uses references to properties of a subform.

Responding to Data Events

`Recordsets` underlying forms and reports trigger data events when you move the record pointer or change the value in one or more cells of the `Recordset`. The most common use of data events is to validate updates to the `Recordset`; you add validation code to the event-handling subprocedure for the BeforeUpdate event. The use of code, rather than setting field-level or table-level `ValidationRule` property values, is that VBA provides a

Oracle Ex. 1009, p. 1099
Oracle v. DeBellis
IPR2024-00221

much more flexible method of assuring data consistency. Validation rules you write in VBA commonly are called *business rules*. Business rules often are quite complex and require access to multiple lookup tables—some of which may be located in other databases.

◀◀ See "Validating Data Entry," p. 170



Listing 30.3 shows an example of a set of validation rules for postal codes in the Suppliers table of Northwind.mdb, the `Recordset` of which is bound to the Suppliers form. The `BeforeUpdate` event, which triggers before a change is made to the `Recordset`, includes a predefined `Cancel` argument. If you set `Cancel = True` in your event-handling code, the proposed update to the `Recordset` does not occur.

Listing 30.3 A VBA Validation Subprocedure for International Postal Codes

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
' If number of digits entered in PostalCode text box is
' incorrect for value in Country text box, display message
' and undo PostalCode value.

   Select Case Me!Country
      Case IsNull(Me![Country])
         Exit Sub
      Case "France", "Italy", "Spain"
         If Len(Me![PostalCode]) <> 5 Then
            MsgBox "Postal Code must be 5 characters", 0, _
               "Postal Code Error"
            Cancel = True
            Me![PostalCode].SetFocus
         End If
      Case "Australia", "Singapore"
         If Len(Me![PostalCode]) <> 4 Then
            MsgBox "Postal Code must be 4 characters", 0, _
               "Postal Code Error"
            Cancel = True
            Me![PostalCode].SetFocus
         End If
      Case "Canada"
         If Not Me![PostalCode] Like _
            "[A-Z][0-9][A-Z] [0-9][A-Z][0-9]" Then
            MsgBox "Postal Code not valid. " & _
               "Example of Canadian code: H1J 1C3", _
               0, "Postal Code Error"
            Cancel = True
            Me![PostalCode].SetFocus
         End If
   End Select
End Sub
```

Data events of combo boxes commonly are used to select a specific record or set of records for display in a form or subform. The EditProducts form of Solutions.mdb, shown in Figure 30.31, displays product and supplier data for selections made consecutively in the Select Category and Select Product drop-down lists. When you make a selection in

Oracle Ex. 1009, p. 1100
Oracle v. DeBellis
IPR2024-00221

the Select Category list, the action populates the Select Product list with products. Selecting a product in the Select product list displays product and supplier data in the text boxes of the form.

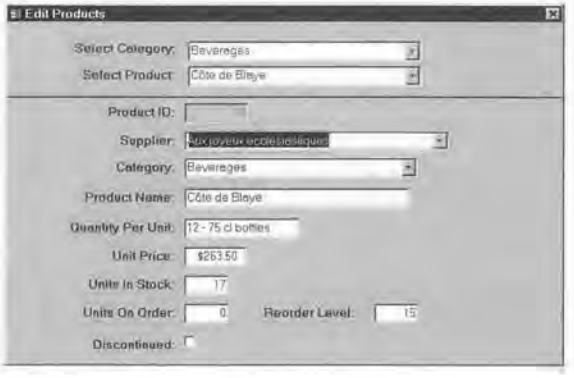


FIG. 30.31 The EditProducts form of the Developer Solutions database (Solutions.mdb).

The VBA code of Listing 30.4, from the class module for the EditProducts form of the Developer Solutions database, illustrates extensive use of data events of the `Form` object and combo box control objects. Making a selection in the Select Category list executes the `SelectCategory_AfterUpdate` event handler, which requeries the Select Product list to display the products for the selected category. Making a selection in the updated Select Product list executes the `SelectProduct_AfterUpdate` event handler, which applies a filter to the `Recordset` of the form to display the product data. In turn, changing the record pointer of the form's Recordset executes the `Form_Current` event handler, which tests for low product inventory.

Listing 30.4 Data Event Handlers in the Class Module for the EditProducts Form of Solutions.mdb

```
Option Compare Database   'Use database order for string comparisons.
Option Explicit

Private Sub Form_AfterUpdate()
   'Requery SelectProduct combo box.
   Me!SelectProduct.Requery
End Sub

Private Sub Form_Current()
'If value in UnitsInStock text box is less than value in ReorderLevel
'text box and value in UnitsOnOrder text box is 0, display
'value in UnitsInStock in red and display LLowStock label.

   If (Me!UnitsInStock + Me!UnitsOnOrder) <= Me!ReorderLevel Then
      Me!UnitsInStock.ForeColor = 255
      Me!LLowStock.Visible = True
   Else
     'Otherwise, display value in UnitsInStock text box in black
```


Oracle Ex. 1009, p. 1101
Oracle v. DeBellis
IPR2024-00221

```
      'and hide LLowStock label.
       Me!UnitsInStock.ForeColor = 0
       Me!LLowStock.Visible = False
   End If
End Sub

Private Sub GoToSupplierID_Enter()
'Go to SupplierID combo box to prevent advancing
'to another record.

   Me!SupplierID.SetFocus
End Sub

Private Sub ReorderLevel_AfterUpdate()
'Run Form_Current procedure.

    Form_Current
End Sub

Private Sub SelectCategory_AfterUpdate()
'Enable and requery SelectProduct combo box.
'Disable controls in detail section.

   Me!SelectProduct.Enabled = True
   Me!SelectProduct.Requery
   EnableControls Me, acDetail, False
End Sub

Private Sub SelectProduct_AfterUpdate()
'Find record for product selected in SelectProduct combo box.
'Enable controls in detail section and disable ProductID text box.
'Go to SupplierID combo box.

    DoCmd.ApplyFilter , "ProductID = Forms!EditProducts!SelectProduct"
    EnableControls Me, acDetail, True
    Me!ProductID.Enabled = False
    Me!SupplierID.SetFocus
End Sub

Private Sub SelectProduct_BeforeUpdate(Cancel As Integer)
   If IsNull([SelectProduct]) Then
      MsgBox "You must select a product."
      Cancel = True
   End If
End Sub

Private Sub UnitsInStock_AfterUpdate()
'Run Form_Current procedure.

    Form_Current
End Sub

Private Sub UnitsOnOrder_AfterUpdate()
'Run Form_Current procedure.

    Form_Current
End Sub
```

Oracle Ex. 1009, p. 1102
Oracle v. DeBellis
IPR2024-00221

The VBA code examples in this chapter only cover the basics of the use of VBA 5.0 for responding to events triggered by forms, controls, and Recordsets bound to forms or reports. A full course in VBA programming exceeds both the scope and the publishing limitations of this book. Many of the examples of this chapter are drawn from the sample databases supplied with Access 97. You can adapt many of the techniques illustrated in the event-handling subprocedures of the sample databases to custom applications you create. To become an expert in VBA programming requires study, experimentation, and perseverance. Periodicals, books, and Web sites, such as those listed in the "Bibliography" section of the introduction to this book, are likely to satisfy the studious reader. There's no substitute, however, for experimentation. Writing and testing code is the only sure way to become proficient in VBA programming.

Oracle Ex. 1009, p. 1103
Oracle v. DeBellis
IPR2024-00221

Chapter 31

Exchanging Data with Automation and ActiveX Controls

Interprocess communication (IPC) has come a long way from its early origins in Windows dynamic data exchange (DDE). DDE made it possible to transfer data between Windows applications by setting up a DDE conversation between the two applications using the Windows Clipboard as an intermediary. Windows 3.1 made Object Linking and Embedding (OLE) 1.0 the preferred method of transferring data between OLE-enabled, 16-bit applications. These transfers are still made via the Clipboard with the Edit, Paste Special menu choice. OLE 2.x, implemented in 16-bit Access 2.0 and 32-bit Access 95, brought drag-and-drop operation to Clipboard-based OLE. OLE 2.x also introduced OLE Automation and programmable objects—the foundation for a Windows component architecture.

Access 2.0 was the first Microsoft application to support 16-bit OLE Controls (OCX), the intended replacement for Visual Basic Extension (VBX) custom controls. OLE Controls let Microsoft and third-party suppliers create special-purpose components to supplement the standard set of native controls offered by Access. Access 95 and Visual Basic 4.0 introduced a common set of 32-bit OLE Controls. Developers found creating OLE Controls with C++ to be a daunting task, and OCXs turned out to be much larger than their VBX counterparts. Despite these hurdles, a wide variety of OLE Controls became available for Access 95 and Visual Basic 4.0.

Microsoft's decision to "embrace and extend the Internet," announced on December 9, 1995, caused Microsoft to rethink its original OLE Control design specification. OLE Controls were much too large to download from the Internet on an as-required basis. Microsoft announced at its Professional Developers Conference on March 12, 1996, the specification for a lightweight version of OLE Controls—called ActiveX Controls—as the major element of a broader ActiveX Technologies program. 32-bit ActiveX Controls are 30 percent to 50 percent smaller than the 16-bit and 32-bit OLE Controls they replace.

This chapter introduces you to (OLE) Automation and ActiveX controls with emphasis on their implementation in Access 97. Automation is the

Oracle Ex. 1009, p. 1104
Oracle v. DeBellis
IPR2024-00221

technology that lets you program Jet's Data Access Objects (DAO) with VBA. Automation objects expose only properties and events. ActiveX Controls extend Automation objects with events that connect to VBA event-handling subprocedures.

Introducing Microsoft's ActiveX Technology

OLE and ActiveX technologies both are built on Microsoft's *Common Object Model* (COM). COM is a specification for designing reusable software components (objects) that you can combine to create useful applications, such as Access database front-ends. The COM specification defines the connection (a set of *interfaces*) between a client, such as Access, and a programmable object—typically the DAO for Access clients. Microsoft's implementation of COM is for Windows; third parties currently are porting COM to the Macintosh and UNIX operating systems. COM objects and their clients must be located on the same computer. *Distributed COM* (DCOM), which is included in Windows NT 4.0, gives clients access to server-based COM objects via a networked connection.

Note

The preliminary specification for COM is available at **http://www.microsoft.com/intdev/sdk/docs/com/comintro-f.htm**. You can read the first two chapters, which cover the basics of COM, in HTML format or download the entire specification in Word 6.0 format.

Microsoft's ActiveX Web site at **http://www.microsoft.com/activex/default.htm** provides links to a wide variety of information on ActiveX Controls, including many useful controls you can download and use in your Access applications.

Microsoft's goal in developing its ActiveX Technologies is to transform COM and DCOM into public Internet standards. In early October 1996, Microsoft announced that the COM, DCOM, and Automation specifications henceforth are to be managed by the Open Group, an industry standards organization. OLE, which deals primarily with embedding and linking documents, remains Microsoft's proprietary technology. Other COM-based Windows technologies over which Microsoft intends to retain control, such as OLE DB (OLE Databases) and OLE DS (OLE Directory Services) also carry the OLE prefix.

Note

The OLE DB specification defines a generalized method for accessing any data that can be formatted into a row-column structure. ActiveX Data Objects (ADO) is a lightweight Automation interface to OLE DB data providers. The first application for ADO is to support Active Database Components in the new ActiveX Server extensions (code-named Denali) to Microsoft's Internet Information Server. Microsoft says in its Frequently Asked Questions for ODE DB that ADO is an "evolution of our earlier object models," and adds, "Over time, all our products will standardize on ADO and OLE DB." You can get more information on OLE DB and ADO at **http://www.microsoft.com/oledb/**.

In conjunction with the transfer of specifications to the Open Group, Microsoft removed the OLE prefix from the names of Internet-related components. Table 31.1 lists the old

Oracle Ex. 1009, p. 1105
Oracle v. DeBellis
IPR2024-00221

and new terms for the affected components and technologies. Microsoft didn't change OLE Automation to ActiveX Automation because COM's Automation interface also is used for programming linked and embedded OLE objects.

Table 31.1 New COM and DCOM Terminology for OLE Technologies

Old Term	New Term
NetworkOLE	Distributed COM (DCOM)
OLE Automation	Automation
OLE Automation servers	ActiveX components
OLE components	ActiveX components
OLE Controls	ActiveX Controls
OLE Designers	ActiveX Designers
OLE document objects	ActiveX documents
OLE servers	ActiveX components
OLE specification	ActiveX specification

The Access 97 documentation continues to use the terms *OLE server* and *OLE container*. The name of Jet's OLE Object field data type does not change because OLE Object fields are intended as containers for linked or embedded OLE objects.

Understanding Automation

Access 97, like Access 95, is an OLE 2.1 client and server application; Access 2.0 was an OLE 2.01 client only. The Jet 3.5 database engine is a programmable Automation object, so VBA-enabled applications, including all members of Office 97, can manipulate the Jet 3.5 DAO with code similar to the Access VBA examples in Chapter 29, "Understanding the Data Access Object Class."

◀◀ See "Writing a Function that Uses Database Objects," 1041



Referencing the Jet 3.5 engine directly in other VBA-enabled applications, rather than using Access 97 as an Automation server, is much faster and uses fewer resources. Therefore, your primary use of Access 97 is likely to be as an Automation client that manipulates various Automation servers, such as Word 97 or Excel 97. The sections that follow describe the types of Automation servers defined by the OLE 2.1 specification and how 16- and 32-bit Automation components interact.

Categorizing Automation Components

Understanding Automation requires an explanation of the categories of Automation components that you can use in building desktop database applications. Automation components are of five basic types:

- *Full servers* are programs, such as 32-bit Microsoft Excel 97 and Word 97, that are stand-alone productivity applications with Automation capability added. Full servers create insertable objects that you display in Access's bound or unbound object frames. Like Access 97, Excel 97 and Word expose their application objects to their

Oracle Ex. 1009, p. 1106
Oracle v. DeBellis
IPR2024-00221

own flavor of VBA through a type library. Full servers also are called *local servers* because the server must reside on the same PC as the Automation client application.

- *Automation servers* are full (local) servers that are not insertable objects. Access 97 is an Automation server only. Access 97 does not appear in the Object Type list of the Create New page of the Insert Object dialog of Word 97 or Excel 97. If you attempt to specify an .mdb file in the Create From File page, Object Packager attempts to create a package from the .mdb file.
- *Mini-servers* are applications, such as Microsoft Graph 8.0 (MSGraph8), that can only be executed from within an Automation client application. Mini-servers are similar to the OLE 1.0 applets originally supplied with Word for Windows 2.0 and other early Microsoft OLE 1.0 applications, such as Microsoft Draw. OLE 2.x mini-servers expose objects through a type or object library.

  To qualify for mini-server status, the application must be an executable file (.exe) that is capable of displaying a window. Mini-servers that display a particular class of objects, such as video clips, are called *viewers*. Using the Graph Wizard to create MSGraph8 charts is one of the subjects of Chapter 20, "Adding Graphics to Forms and Reports." You also can create custom graphs and charts by programming MSGraph8 with Access VBA code.
- *ActiveX Controls (OCXs)* are a special type of mini-server. ActiveX Controls, which use the .ocx file extension, expose events in addition to properties and methods. They are the object-oriented equivalent of the VBX custom controls of Visual Basic 3.0 and earlier. Most ActiveX Controls have a visible user interface. Access 97 includes a single ActiveX Control—the Calendar Control (Msacal.ocx). The Office Developer Edition (ODE) includes a variety of useful ActiveX Controls and a license to distribute the Controls with your Access applications.



▶▶ See "Examining the Content of the Office 97 Developer Edition," p. 1135

- *Process servers* are a subclass of automation servers. Process servers are used to perform functions that do not involve interaction at the user interface. Process servers come in two flavors: OutOfProc(ess) and InProc(ess) servers. *OutOfProc* servers are executable files that run in their own process space; that is, they have their own block of allocated memory. (Full servers and mini-servers are OutOfProc servers.) *InProc* servers share memory with the client application. (ActiveX Controls are InProc servers.)

  OutOfProc servers communicate with local client applications through Lightweight Remote Procedure Calls (LRPCs), while InProc servers, also called OLE DLLs, use conventional Windows function calls. Thus InProc servers respond considerably faster to client instructions than OutOfProc servers. The advantage of OutOfProc servers is the ability to locate the server on a remote computer and communicate

Oracle Ex. 1009, p. 1107
Oracle v. DeBellis
IPR2024-00221

with the server via DCOM. In this case, the OutOfProc server is called a *Remote Automation Object* (RAO). Microsoft's recently announced Viper transaction processing technology lets you create in-process RAOs.

Note

Process server is not an "official" OLE or COM category. The term is used here to distinguish invisible programmable process server objects (that run in the background) from programmable objects with visible representation in Access 97's run or design modes, even if the visible representation is optional.

One of the primary applications for process servers is creating "three-tier" client/server database applications. A three-tier application interposes an RAO between the client front end and the RDBMS back end. The RAO manages connecting to the RDBMS and the processing of queries. The RAO may include code that enforces "business rules" common to all front-end applications. The middle-tier RAO isn't visible to the user.

One of the advantages of Automation is that you can manipulate programmable full-server and mini-server objects without creating a visible instance of the OA server. Unlike DDE, which requires that the server application be launched in a window or as an icon, Automation automatically launches the application for you. Unless you instruct the Automation server to activate its window, the server is invisible; the server's name usually does not appear in the Windows Task List.

Note

Although a full or automation server's window isn't visible, full and automation servers can consume a substantial percentage of the available resources of your computer. This fact is particularly true of mega-apps, such as Excel 97 and Word 97, when used as Automation servers. You can speed the opening of Excel 97 as an OA server and minimize the resources Excel 97 consumes by not loading unneeded add-ins.

If you plan to use Automation extensively in your Windows 95 applications, you should have a minimum of 16M of RAM (preferably 24M or more) and make sure that you have plenty of disk space (30M or more) for the Windows 95's swap file. For Windows NT 4.0 Workstation, 24M of RAM and a paging file of 48M or more is a good starting point.

Interacting with 16-Bit OLE Automation Servers

You can use most 16-bit OLE 1.0 and 2+ OutOfProc servers with 32-bit Automation clients. Many developers supply only a 16-bit version of their OutOfProc servers in order to be compatible with Windows 3.1+, Windows 95, and Windows NT 4.0. The rules for using 16-bit and 32-bit programmable objects in both Windows 95 and Windows NT 4.0 are as follows:

- 32-bit OLE 2+ and Automation clients can employ 16-bit OLE 1.0 and 16-bit and 32-bit OLE 2+ full servers and mini-servers. The conversion to and from 16-bit operations is accomplished by a process called *thunking*, which is implemented by Olethk32.dll.

Oracle Ex. 1009, p. 1108
Oracle v. DeBellis
IPR2024-00221

- 16-bit OLE clients cannot employ 32-bit OLE 2+ or Automation servers.
- 32-bit applications that support ActiveX Controls must use 32-bit OCXs.
- 16-bit applications that support early OLE Controls—only Access 2.0 and 16-bit Visual Basic 4.0—must use 16-bit OCXs. Full support for Visual Basic 16-bit OCXs in Access 2.0 requires installation of several updated files. (ActiveX Controls are 32-bit only.)
- InProc servers (OLE DLLs) must have the same "bitness" as the client application. As an example, you cannot use the 16-bit Jet 2.x database engine with Access 97 or the 32-bit Jet 3.x database engine with Access 2.0.

Note

The thunking process for IPC between 32-bit OLE 2+ clients and 16-bit OLE 1.0 and 2+ servers imposes additional overhead on LRPCs. This process also extracts an application performance toll. If you intend to make extensive use of Automation with Access 95, an investment in 32-bit versions of Automation servers will return substantial performance dividends.

On the whole, the limitations associated with the interoperability of 16-bit and 32-bit OLE Automation components are unlikely to be a problem for Access 97 users running either Windows 95 or Windows NT 4.0. The vast majority of Microsoft and third-party Windows application development has been devoted to 32-bit Windows software since early 1995. It is unlikely that software publishers will make major upgrades to their existing 16-bit applications for use with Windows 3.1+.

Using Automation Servers with Access 97

To use OLE Automation, you must first write the code to create an instance in your application of the object that you plan to program. Access VBA contains the following four reserved words that you use to create an instance of a programmable object with Access VBA code:

- The **Object** data type is assigned to the programmable object variables you declare with {**Private**¦**Dim**¦**Public**} obj*Name* **As Object** statements. The `Object` property of a programmable Automation object contained in a bound or unbound object frame control points to the instance of the object. You can assign to an **Object** variable a pointer to the Automation object with the general syntax

```
Set objName = [Forms!frmName!]uofName.Object
```

- The `CreateObject` function assigns a pointer to a new instance of an empty programmable object, such as a blank Excel 97 `Worksheet` object. The general syntax of the `CreateObject` function is

```
Set objName = CreateObject("ServerName.ObjectType")
```


Oracle Ex. 1009, p. 1109
Oracle v. DeBellis
IPR2024-00221

- The `GetObject` function assigns a pointer to a new instance of a programmable object whose data is contained in an existing file, *strPathFileName*, in the following syntax example:

```
Set objName = GetObject(strPathFileName[, _
   "ServerName.ObjectType"])
```

  You can omit the *ServerName.ObjectType* argument if an entry in the Registry associates the file's extension with the object of the application you want to program. In the case of Excel 97, the default object type for .xls files is `Excel.Workbook`. If you substitute an empty string (`""`) for *strPathFileName*, the preceding statement assigns a pointer to an object of *ServerName.ObjectType* if such an object is open. If an object of the specified type is not open when the statement is executed, a trappable error occurs.

- The **New** keyword declares an **Object** variable and assigns in a single command a pointer to a new instance of an empty programmable object, such as an empty Word 97 document. The general syntax for creating an instance of a programmable object with **New** is

```
{Private|Dim|Public|Static} objName As New [ServerName.]ObjectType
```

  To use the **New** keyword to create an instance of an **Object** variable, you must add a reference to the Automation server in Access 97's References dialog. Using the **New** keyword to create an object instance is called *early binding*. The advantage of early binding is that the VBA interpreter can check the validity of object references in Design view and the new Auto List Members feature displays candidate objects in a popup list (see Figure 31.1). Auto List Members also displays the syntax for object properties and methods.

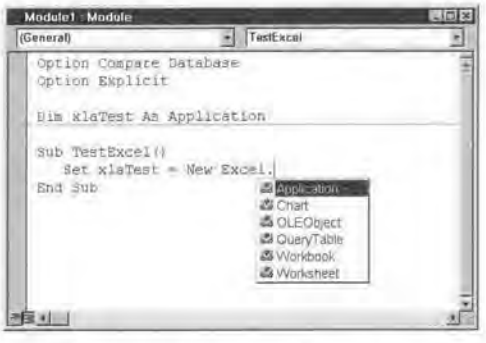


FIG. 31.1 The Auto List Members feature displaying the first-tier objects of the Microsoft Excel 8.0 type library.

If the Automation server application is open when you use the `CreateObject` or `GetObject` functions or the **New** reserved word, the type of the Automation server determines whether a new instance of the server is launched or the open instance is used. If the

Oracle Ex. 1009, p. 1110
Oracle v. DeBellis
IPR2024-00221

server type is Creatable Single-Instance, as is the case for Word 97, the open instance of Word is used with an empty document (`CreateObject`) or with a document created from the specified file (`GetObject`).

> **Note**
>
> Names of programmable objects created by Automation server applications appear in regular (not bold), monospaced type because the names of these objects are not VBA object data types.
>
> This book uses the common lowercase file extensions as the prefix for identifying objects created by full servers (rather than using the object tags specified by the *Leszynski Naming Conventions for Microsoft Access (LNC)*, incorporated as Appendix A, "Naming Conventions for Access Objects and Variables"). Examples are `xlaApp` for the Excel application, `xlwBook` for Excel 5+ workbooks, `xlsSheet` for worksheets, `xlcChart` for Excel charts, and `xlmModule` for Excel VBA modules. (LNC uses xlsa*App*, xlsw*Sheet*, and xlsc*Chart*; LNC does not include a tag for Excel workbooks or VBA modules.) Word 97 `Document` objects are identified in this book by wdd*Name*, and the Word `Application` object is wda*Name*.

The four reserved words of the preceding list also are used by VBA to program objects of other applications. To write Automation code, you need to know the server name of the OA server and the names of the object types created by the server. The best way to become familiar with the object hierarchy of an Automation server is to create a reference to the server and explore the object hierarchy with Access 97's Object Browser.



◀◀ See "Using the Object Browser," p. 1027

Manipulating an Excel 97 *Workbook* Object

The hierarchy of Excel 97 programmable objects is at least as complex as that of Access 97. Fortunately, the concepts of addressing objects in all Automation-compatible applications is nearly identical. Thus, the techniques you learned in Chapter 29, "Understanding the Data Access Object Class," stand you in good stead when you encounter Excel 97's `Excel.Application` object. The following sections describe how to create a test workbook, Customers.xls, with a single worksheet, Customers, and how to transfer data to and from the worksheet using Access VBA and Automation.

Creating Customers.xls. The examples that follow use the Customers.xls workbook file with data from the Customers table of Northwind.mdb. The example of OLE Automation with Excel 97 in this chapter uses the `GetObject()` function to open the Customers.xls file and manipulate the data in the worksheet. To create Customers.xls, follow these steps:

1. Select the Customers table in the Database window, and click the selection arrow of the Office Links button. Choose Analyze It with MS Excel from the drop-down menu to export the data in the table to Customers.xls and open the file in Excel 97 (see Figure 31.2).

Oracle Ex. 1009, p. 1111
Oracle v. DeBellis
IPR2024-00221

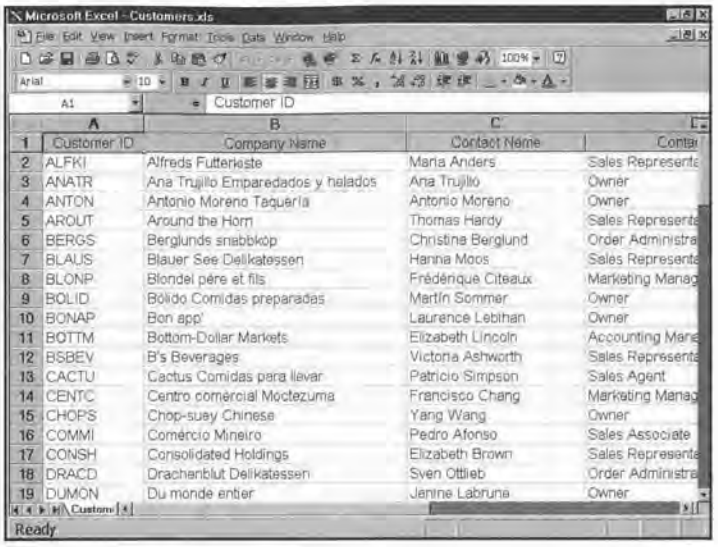


FIG. 31.2 The data of the Customers table exported to a Microsoft Excel 97 workbook.

2. Select cells A2 through D9 of the worksheet. Choose Insert, Name, and select Define. Type TestRange in the Names in Workbook text box and click OK. This creates a named range that you use in the examples that follow.

3. Choose File, Save As from Excel's File menu, and select Microsoft Excel Workbook in the Save As Type drop-down list to save the file in the default Microsoft Excel 5.0/95 (.xls) format. Alternatively, you can save the file in Excel 97's default Excel 8.0 format.

4. Change the Save In folder to your \Program Files\Microsoft Office\Office\Samples folder, and accept the default name for the file, Customers.xls, unless you have a reason to do otherwise. Click Save to save the file and close the dialog.

6. Close Microsoft Excel.

The Hierarchy of VBA Excel 95 Objects. The following list describes the hierarchy of the most commonly used programmable objects exposed by Excel 97:

- `Application` represents an invisible instance of Excel 97. Using the `Application` object, you can execute almost all of Excel's menu commands by applying methods to the `Application` object. The `Application` object has properties such as `ActiveWorkbook` and `ActiveSheet` that specify the current `Workbook` and `Worksheet` objects. You can specify `Excel.Application` as the value of the `ServerName`. `ObjectType` argument of the `CreateObject()` and `GetObject()` functions, plus the `Dim` *`objName`* `As New Excel.Application` statement.

- `Workbook` is the primary persistent object of Excel 97. `Workbook` objects are files that contain the other objects you create with Excel 97: `Worksheet` and `Chart` objects. `Worksheet` and `Chart` objects are contained in `Worksheets` and `Charts` collections, respectively.

Oracle Ex. 1009, p. 1112
Oracle v. DeBellis
IPR2024-00221

- `Worksheet` objects are elements of the `Workbook` object that contain data. The primary interaction between Access 97 and Excel 97 takes place with `Worksheet` objects. You can transfer data contained in rows and columns of an Access `Recordset` object to cells in an Excel `Worksheet` object and vice versa. If you specify `Excel.Sheet` as the value of the *`ServerName.ObjectName`* argument of the `GetObject` function or use the `Dim` *`objName`* `As New Excel.Sheet` statement, the first member of the `Worksheets` collection—the `ActiveSheet` property of the `Workbook` object—is opened by default.

- `Range` objects are groups of cells specified by sets of cell coordinates. You can get or set the values of a single cell or group of cells by specifying the `Range` object. If the cells are contained in a named range, you can use the range name to refer to the group of cells. (A group of cells is not the same as a collection of cells; no `Cells` collection exists in Excel 97.)

  The `Cells` method specifies the coordinates of a `Range` object. Although no `Cell` object exists in Excel 97, you can use the `Cells` method to read or set the value of a single cell or group of cells, specified by coordinate sets.

Excel 97 exposes a variety of other objects. The preceding four objects, however, are those that you use most often in conjunction with Automation operations that are executed with Access VBA code.

Using Excel's Online Help File for VBA. If you don't have a printed copy of the *Microsoft Excel Visual Basic for Applications Reference*, you can open the Vbaxl8.hlp online help file to act as a reference for the objects, methods, and properties exposed by Excel 97. To open Vbaxl8.hlp while running Access and to display objects and methods references, follow these steps:

1. Launch Explorer and select your Excel 97 folder (usually C:\Program Files\Microsoft Office\Office). Double-click Vbaxl8.hlp to open the Help Topics: Microsoft Excel Visual Basic dialog's Contents page (see Figure 31.3).

2. Double-click the Microsoft Excel Visual Basic Reference chapter icon, then double-click the Microsoft Excel Objects icon to display a diagram of the hierarchy of objects exposed by Excel 97 (see Figure 31.4). Objects exposed to Excel VBA are exposed also to Automation client applications.

3. Click the Worksheets (Worksheet) hotspot of the diagram to display the help topic for the `Worksheets` Collection Object, and click the Worksheet hotspot to display the help topic for the `Worksheet` Object.

4. Click the Properties hotspot to display the list of proprieties of the `Worksheet` object in the Topics Found dialog (see Figure 31.5). You can obtain a similar list of methods that apply to the object by clicking the Methods hotspot. (`Cells` was a method of Excel 95's Worksheet object.)

5. Double-click the Cells Property item hotspot to display the help window for the `Cells` property (see Figure 31.6).

Oracle Ex. 1009, p. 1113
Oracle v. DeBellis
IPR2024-00221



FIG. 31.3 The Contents page of the Vbaxl8.hlp file.

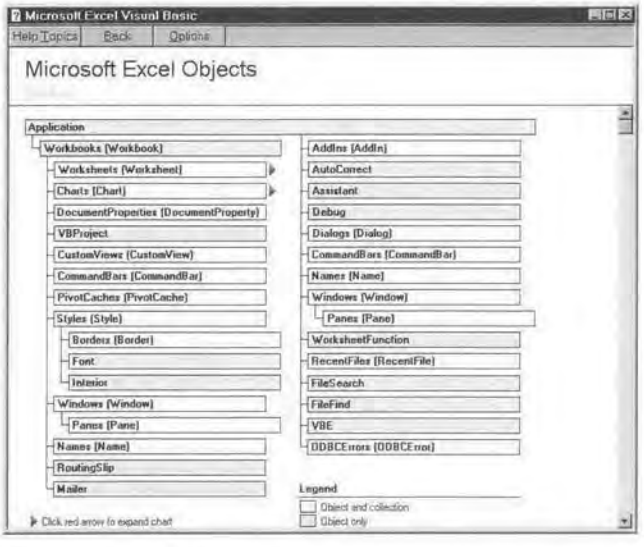


FIG. 31.4 The Microsoft Excel Objects hierarchy displayed by the Excel VBA help file.

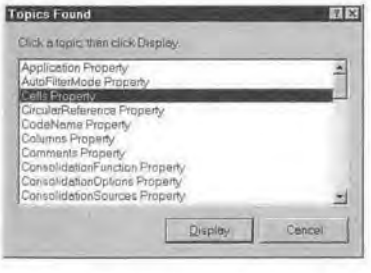


FIG. 31.5 Properties of the Excel `Worksheet` object displayed in the Topics Found window.

Oracle Ex. 1009, p. 1114
Oracle v. DeBellis
IPR2024-00221

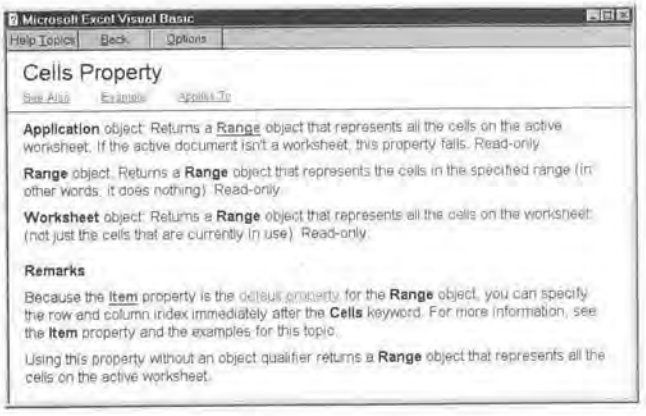


FIG. 31.6 The help topic for the `Cells` property of the `Application`, `Range`, and `Workbook` objects.

6. Click the Examples hotspot to display examples of VBA code applicable to the `Cells` property (see Figure 31.7). Most of the Excel VBA code examples are compatible with Access VBA Automation code.

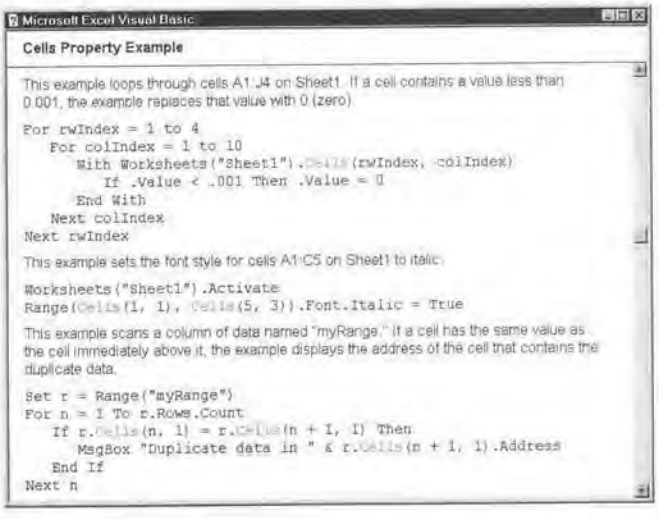


FIG. 31.7 VBA example code for the `Cells` property of the `Worksheet` and `Range` objects.

Creating an Excel Reference and Using Access 97's Object Browser. Creating a reference to the object or type library of Automation servers offers the following advantages:

- The Access VBA compiler checks the syntax of your Automation code for consistency with the properties and methods of the objects you declare. This test is not made if you don't declare a reference to the Automation server object.

Oracle Ex. 1009, p. 1115
Oracle v. DeBellis
IPR2024-00221

- You can use the Object Browser to determine the proper syntax for object properties and methods.
- You can gain quick access to the help topic for the object, property, or method selected in the Object Browser's lists.
- You gain the advantage of Access 97's Auto List Members feature when writing your Automation code.
- You can use the shorthand `Dim objName As New ServerName.ObjectType` statement to instantiate (create an instance of) the server object with early binding.
- You can use the intrinsic constants predefined for the object as argument values. This is especially important for servers, such as Excel, that have many intrinsic constants.

To create a reference to the Microsoft Excel 8.0 object library and explore Excel objects with the Access 97 Object Browser, follow these steps:

1. Launch Access and open Northwind.mdb, if necessary. Open the Utility Functions module and choose Tools, References to display the References dialog.
2. Mark the check box for the Microsoft Excel 8.0 Object Library, Excel8.olb (see Figure 31.8), and then click the OK button to close the References dialog and create the reference.

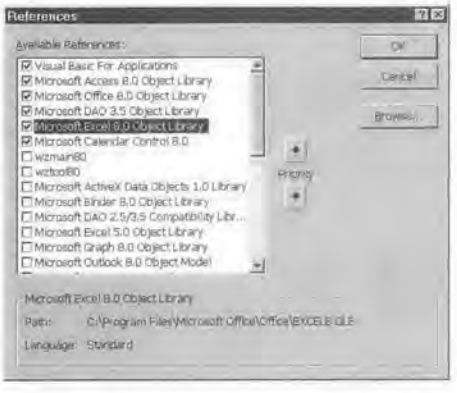


FIG. 31.8 Adding an Access VBA reference to the Microsoft Excel 8.0 Object Library.

3. Open Access's Object Browser and choose Excel in the Project/Library list (see Figure 31.9).
4. Select Worksheet in the Classes list and Cells in the Members of 'Worksheet' list. A cryptic syntax example for the `Cells` property appears in the bottom panel. Clicking the Help button displays the help topic for the `Cells` property, which is the same topic shown in Figure 31.7.

Oracle Ex. 1009, p. 1116
Oracle v. DeBellis
IPR2024-00221



FIG. 31.9 Displaying the simplified syntax for the `Cells` property of the `Worksheet` object in Access 95's Object Browser.

5. When you finish exploring properties and methods of the Worksheet object, click the Close button to close Object Browser.

Registry Entries for Automation Servers. You can specify any object listed in the Registry that is identified by a *`ServerName.ObjectType`*`[.Version]` entry at the HKEY_CLASSES_ROOT level of the Registry, as shown in Figure 31.10. The optional *`.Version`* suffix, shown in the line below the entry for `Excel.Sheet`, indicates the version number of ServerName. If you have two versions of the same Automation server, you can use the `.Version` suffix to specify one of the two.

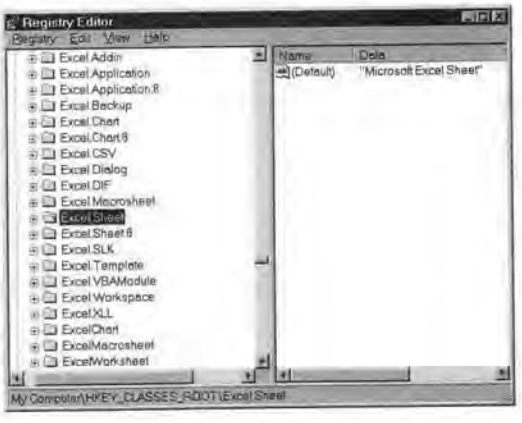


FIG. 31.10 Registry entries for top-level Excel 97 objects that you can call directly from Automation clients.

In addition to the primary Registry entries for Excel 97 objects that you can open, additional data for creating or opening programmable objects appear in the `\HKEY_LOCAL_MACHINE\Software\CLSID` section of the Registry. `CLSID` is the abbreviation for `ClassID`, a globally unique, 32-character (plus hyphens) identifier for each object exposed

Oracle Ex. 1009, p. 1117
Oracle v. DeBellis
IPR2024-00221

by any COM object. The additional Registry entries for the `Excel.Sheet.5` object appears in Figure 31.11. These entries provide information required to launch Excel 97 in Automation mode when you create an `Excel.Sheet` object.



FIG. 31.11 Additional registration database entries for Excel 95 `Application` and `Worksheet` objects.

Opening and Manipulating an Existing Excel Worksheet Object. Learning to write VBA Automation code is a much quicker process if you use the Debug window to try Automation instructions before you begin writing complex Automation code in modules.

> **Note**
>
> In all examples of this chapter in which the Debug window is used, pressing the Enter key after typing **? *Expression*** is implied.

To open the Worksheet object of the Customers.xls Workbook object, follow these steps:

1. Close Excel if it is running. Open a new module in the Northwind Traders database, and add the following Object variable declarations in the Declarations section:

```
Private xlaCust As Object 'Application object
Private xlwCust As Object 'Workbook object
Private xlsCust As Object 'Sheet object
```

2. Open the Debug window and type the following statement to create an object of the default `Workbook` class. When you press Enter, Excel 97 is launched in `/automation` mode.

```
Set xlwCust = GetObject(CurDir & "\customers.xls")
```

`CurDir` returns the well-formed path of your current directory, \Program Files\Microsoft Office\Office\ \Samples, if you've opened Northwind.mdb in its default

Oracle Ex. 1009, p. 1118
Oracle v. DeBellis
IPR2024-00221

location. If you saved Customers.xls elsewhere, add the path to the file name in the preceding statement. Depending on the speed of your computer, opening Excel 97 may take an appreciable period. (When `<Running>` returns to `<Ready>` in the status bar of the Debug window, Excel has finished loading.)

3. Customers.xls only includes one `Worksheet` object, so the Customer worksheet is the `ActiveSheet` object. To open the Customers worksheet, type

   Set xlsCust = xlwCust.ActiveSheet

4. Verify that you have a valid `Worksheet` object by typing **? xlsCust.Cells(1, 1)**. After a brief interval, the expected result—Customer ID—appears.

5. You can test the ability of the `Cells` method to return the values of other cells by typing **? xlsCust.Cells(R, C)**, where **R** is the row and **C** is the column of the cell coordinates (see Figure 31.12).



FIG. 31.12 The commands to read and set the values of a single cell in the Customers worksheet.

6. You can alter the content of a cell by entering an expression such as `xlsCust.Cells(2, 2).Value = "Alfred's Food Store"`. Like many Access control objects, in which the name of an object returns its value, the `Cells` method does not require that you explicitly specify the default `Value` property. Verify that the content changed by typing **? xlsCust.Cells(2, 2)** with and without appending **.Value** (see Figure 31.12).

7. When you are finished using an Automation object, you should close the object to free the memory resources it consumes. To disassociate an object from an object variable, use a **`Set` `objName` `= Nothing`** statement (refer to Figure 31.12). Type **Set xlsCust = Nothing**, and **Set xlwCust = Nothing** to close the invisible instance of Excel. Multiple object variables can point to a single object, so the OA object is not closed until all object variables have gone out of scope or **`Set`** explicitly to **`Nothing`**.

Oracle Ex. 1009, p. 1119
Oracle v. DeBellis
IPR2024-00221

You also can use the Formula property to set the value of a cell. The advantage of the Formula property is that you can also use this property to enter a formula using Excel A1 syntax, such as =A10+B15.

Using Named Ranges of Cells. If you created the named range—TestRange—in Customers.xls, you can return the values of the cells in the range by referring to the Range object of the Worksheet object. Figure 31.13 shows typical expressions that operate on the Range object of a Worksheet object and the Names collection of a Workbook object.

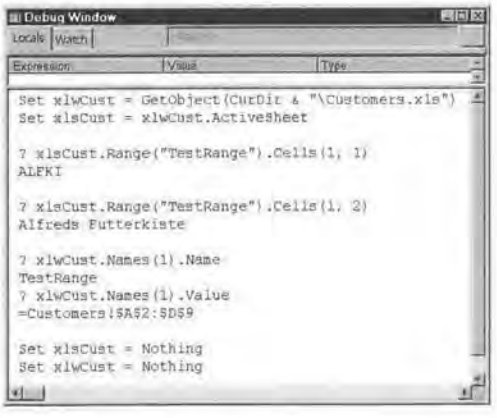


FIG. 31.13 Expressions that return the values in Range objects and the Names collection.

To specify a cell within a named Range object, use the following general syntax:

```
wksName.Range(strRangeName).Cells(intRow, intCol)
```

Because named ranges are global for all worksheets in an Excel workbook, the Names collection is a member of the Workbook object.

You refer to the Ranges collection of the Workbook object with statements such as

```
strRangeName = xlwName.Names(intIndex).Name
strRangeValue = xlwName.Names(intIndex).Value
```

Unlike Access object collections, which begin with an index value of 0, the first member of a VBA collection has an index value of 1.

Explicitly Closing a Workbook and the Application Object. Just as a Workbook object is the value of the Parent property of a Worksheet object, the Application object is the Parent property of a Workbook object. The value of Excel 97's Application object is Microsoft Excel. Figure 31.14 illustrates expressions that work their way upward in Excel's object hierarchy, from Worksheet to Application objects. You can shortcut the process by using xlsCust.Parent.Parent to point to the Application object from a Worksheet object.

You can use the Close method to close an Excel Workbook object explicitly (you can't close a Worksheet object). You can use the Quit method of the Application object to exit the application. It's a good practice to exit the Application server application to conserve

Oracle Ex. 1009, p. 1120
Oracle v. DeBellis
IPR2024-00221

scarce system resources if you have multiple references to the Application server object. (Access 2.0 required brackets around `Close` and `Quit` because of conflicts with Access Basic reserved words.) For example, the following statements close the `Workbook` object and then exit the OA server application, freeing the memory reserved by the server:

```
Set xlaCust = xlsCust.Parent.Parent
xlsCust.Parent.Close(False)
xlaCust.Quit
```

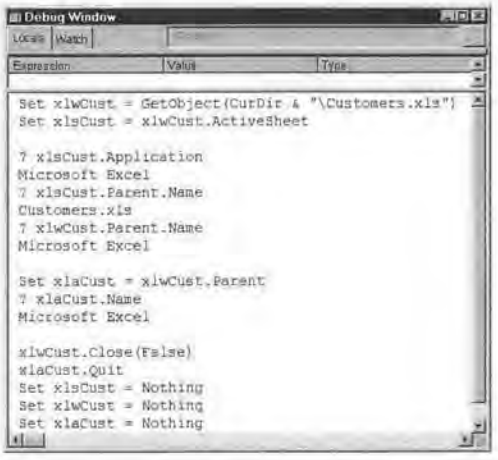


FIG. 31.14 Assigning objects higher in the hierarchy and closing `Workbook` and `Application` objects.

The `False` argument of the `Close` method closes the Workbook without displaying the message box that asks if you want to save changes to the Workbook. When using the `Quit` method to exit the application, you receive that message regardless of whether you modified the worksheet. To avoid this message when reading Excel 95 worksheets, close the worksheet before applying the `Quit` method. You must create an `Application` object because you cannot execute the `xlsCust.Parent.Parent.Quit` statement after closing the `xlsCust` object. (You receive an `Object has no value` error message if you try.)

Creating a New Excel Worksheet with Access VBA Code

You can emulate the Analyze It with MS Excel feature of Access 95 with Access VBA OLE Automation code. The `CreateCust` function of Listing 31.1 creates a new `Workbook` object, Cust.xls, and copies the data from the Customers table to a Worksheet named *Customers*. One of the primary incentives for writing your own version of the Analyze It with MS Excel feature is the ability to format the `Worksheet` object the way that you want. You also can add custom column headers. (The code to add column headers is not included in the `CreateCust` function.) When you execute the function from the Debug window, you receive a progress report from the `Debug.Print` statements in the code.

Oracle Ex. 1009, p. 1121
Oracle v. DeBellis
IPR2024-00221

Listing 31.1 Declarations and Code to Create a New Excel Workbook

```
Option Compare Database
Option Explicit

Private xlaCust As Object 'Application
Private xlwCust As Object 'Workbook
Private xlsCust As Object 'Worksheet

Function CreateCust() As Integer
'Purpose: Create new Excel 95 worksheet from Customers table

   'Declare local variables (Object variables are module-level)
   Dim dbNWind As Database      'Current database
   Dim rstCust As Recordset     'Table Recordset over Customers
   Dim intRow As Integer        'Row counter
   Dim intCol As Integer        'Column counter

   'Assign DAO pointers
   Set dbNWind = CurrentDb()
   Set rstCust = dbNWind.OpenRecordset("Customers", dbOpenTable)

   DoCmd.Hourglass True

   'Create a new Excel Worksheet object
   Set xlwCust = CreateObject("Excel.Sheet.8")
   Set xlsCust = xlwCust.ActiveSheet
   Debug.Print "Worksheet created"

   'Give the new worksheet a name
   xlsCust.Name = "Customers"

   'Get the Application object for the Quit method
   Set xlaCust = xlsCust.Parent.Parent

   intRow = 1
   intCol = 1

   rstCust.MoveFirst   'Go to the first record (safety)
   Do Until rstCust.EOF
      'Loop through each record
      For intCol = 1 To rstCust.Fields.Count
         'Loop through each field
         If Not IsNull(rstCust.Fields(intCol - 1)) Then
            xlsCust.Cells(intRow, intCol).Value = _
            CStr(rstCust.Fields(intCol - 1))
         End If
      Next intCol
      Debug.Print "Record " & intRow
      rstCust.MoveNext
      intRow = intRow + 1
   Loop

   Debug.Print "Formatting columns"
```

(continues)

Oracle Ex. 1009, p. 1122
Oracle v. DeBellis
IPR2024-00221

Listing 31.1 Continued

```
    For intCol = 1 To xlsCust.Columns.Count
       'Format each column of the worksheet
       xlsCust.Columns(intCol).Font.Size = 8
       xlsCust.Columns(intCol).AutoFit
       If intCol = 8 Then
          'Align numeric and alphanumeric postal codes left
          xlsCust.Columns(intCol).HorizontalAlignment = xlLeft
       End If
    Next intCol

    DoCmd.Hourglass False

    Debug.Print "Saving Workbook and exiting"
    xlwCust.SaveAs (CurDir & "\Cust.xls")
    xlaCust.Quit
    Set xlsCust = Nothing
    Set xlwCust = Nothing
    Set xlaCust = Nothing
End Function
```

Note

The `Excel.Sheet` object of Excel 97 returns a `Workbook`, not a `Worksheet` object, when used as the argument of the `CreateObject` function. The `Excel.Sheet` object of Excel 95 returned the expected `Worksheet` object.

Figure 31.15 shows the `Private Object` variable declarations in the Declarations section of the module, followed by the first few lines of the preceding code and entries in the Debug window resulting from executing the `CreateCust` function.

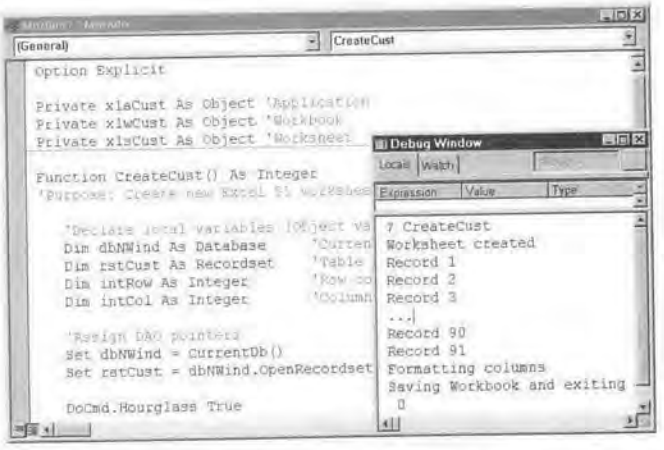


FIG. 31.15 The Declarations section and first few lines of code for the `CreateCust` function.

You must coerce to **`String`** the data type of the **`Variant`** values returned by `rstCust.Fields(intCol -1)` with the **`CStr`** function. If you omit the **`CStr`** function, Excel 97

Oracle Ex. 1009, p. 1123
Oracle v. DeBellis
IPR2024-00221

displays #N/A# instead of the proper value. If the `Recordset` contains field data types other than Text, use the appropriate **`CType`** function to determine the data type for the column.

The `xlLeft` constant, assigned to the value of the `HorizontalAlignment` property of the eighth column, is an Excel intrinsic constant defined when you create a reference to the Microsoft Excel 8.0 Object library. Selecting Constants in the Classes list of the Object Browser with the Excel library active displays the `xlConst` constants. Figure 31.16 shows the numeric value of `xlLeft`, one of the constant values that is valid for the `HorizontalAlignment` property.



FIG. 31.16 Obtaining values of `xlConst` intrinsic constants in Access 95's Object Browser.

Run the function shown in the preceding listing by typing **? CreateCust** in the Debug window. Figure 31.17 shows the Cust.xls workbook with the Customer worksheet created by the `CreateCust()` function opened in Excel 97. Unless you specify a path, Excel saves Cust.xls in the default \My Documents folder.



FIG. 31.17 Part of the Excel 97 worksheet created from the Customers table.

Oracle Ex. 1009, p. 1124
Oracle v. DeBellis
IPR2024-00221

Sending Data to Microsoft Word 97 with Automation

Microsoft Word 97 replaces the venerable Word Basic macro language with VBA 5.0. Word 95 exposed the `Word.Basic` object, which gave VBA programmers access to about 800 Word Basic commands, but Word 95 was an Automation server only. Now Word 97 is a full-fledged Automation server and client with programming capabilities similar to Excel 97. You can create references to type libraries and then use Word VBA to program other application's objects, such as Access 97's `Application` object or the DAO.

> **Note**
>
> If you've programmed the `Word.Basic` object and Word 95 menu commands with Access 95 or Visual Basic 4.0, you find that your VBA client code is backwardly compatible with Word 97. Plan to rewrite Automation code that uses the `Word.Basic` object because there's no assurance that future versions of Word will continue to support the `Word.Basic` object and its Word 95 menu commands.

Using Word 97's Online VBA Help File. Word 97's object model is even more complex than the object model of Excel 97, so you need the Word VBA help file to guide you when using Word as an Automation server. To display the object model for the Word `Application` and `Document` objects, follow these steps:

1. Launch Explorer and select the Word 97 folder (C:\Program Files\Microsoft Office\Office is the default location). Double-click Vbawrd8.hlp to open the Contents page of the Help Topics: Microsoft Word Visual Basic dialog (see Figure 31.18).

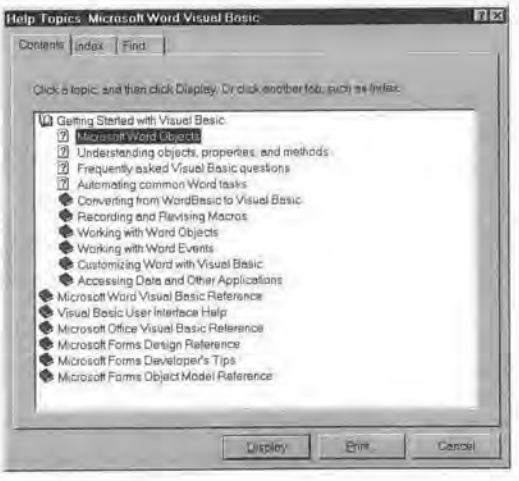


FIG. 31.18 The Contents page of the Vbawrd8.hlp file.

2. Double-click the Getting Started with Visual Basic chapter icon, and then double-click the Microsoft Word Objects page icon to display the top-level (`Application`)

Oracle Ex. 1009, p. 1125
Oracle v. DeBellis
IPR2024-00221

objects exposed by Word 97's object library (see Figure 31.19). Objects exposed to Word VBA are available to all Automation client applications.

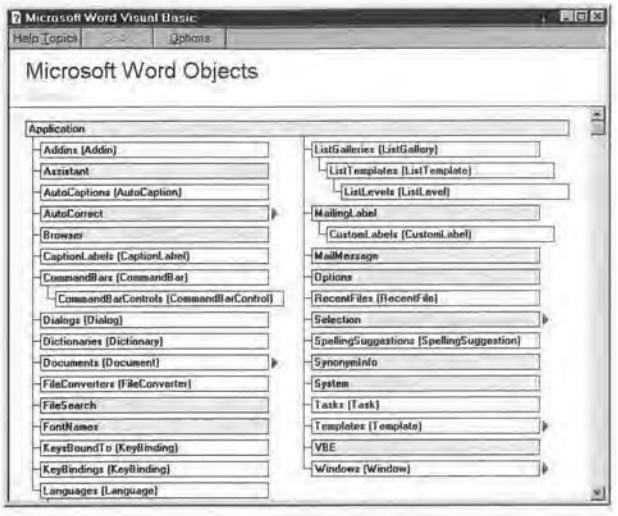


FIG. 31.19 Most of the Microsoft Word Objects hierarchy displayed by the Word VBA help file.

3. Click the Documents (Document) triangular hotspot of the diagram to display the `Document` object model (see Figure 31.20). `Document` objects are the most common entry point for Automation clients that manipulate existing Word documents.

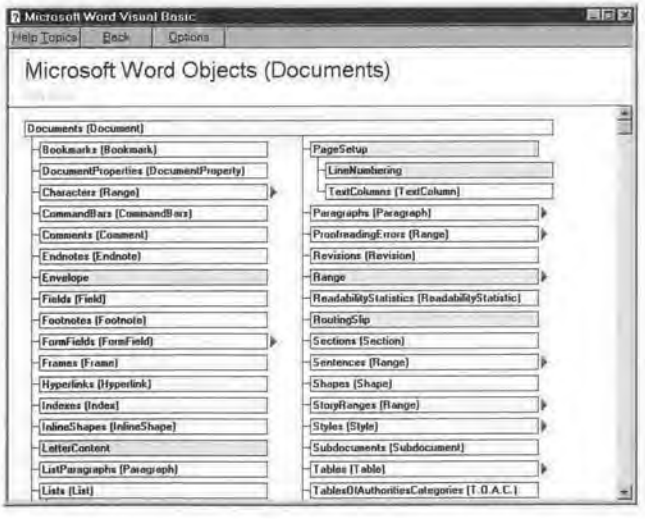


FIG. 31.20 Most of the subobjects of Word 97's Document object.

Oracle Ex. 1009, p. 1126
Oracle v. DeBellis
IPR2024-00221

4. Click the Bookmarks (Bookmark) item to display the help topic for the `Bookmarks` Collection, and then click the Bookmark hotspot to display the help topic for the `Bookmark` object. A `Bookmark` object corresponds approximately to a named `Range` object of Excel.



FIG. 31.21 The Word 97 help topic for the `Bookmark` object.

> **Note**
>
> The `Range` object of a Word document defines a temporary contiguous set of characters in a Word `Document`. Unlike Excel's named `Range` objects, you can only create a Word `Range` object with Word VBA code. Word `Range` objects are not persistent; they exist only for the duration of the procedure that creates the `Range`.

For quick access to help with Word objects and the syntax for properties and methods, create a reference to the Microsoft Word 8.0 Object Library (see Figure 31.22). When you select Word in the Library/Class drop-down list of the Object Browser, Word 97's objects appear in the Classes list. Figure 31.23 shows the properties and methods for the `Bookmark` object in the Members of 'Bookmark' list.

Modifying a Word Document with VBA Code. The generic code for opening an existing Word 97 Document with VBA and saving any changes made to the file is as follows:

```
Private wdaName As Object 'Word Application object
Private wddName As Object 'Word Document object

Set wddName = GetObject("d:\path\filename.doc")
Set wdaName = wddName.Application

'... Code to manipulate document

wdaName.Save
```


Oracle Ex. 1009, p. 1127
Oracle v. DeBellis
IPR2024-00221

```
wdaName.Quit
Set wddName = Nothing
Set wdaName = Nothing
```

If the document includes bookmarks, you can select and replace the bookmarked text with the following code:

```
wddName.Bookmarks("BookmarkName").Select
wdaName.Selection = "Replacement text"
```



FIG. 31.22 Creating a reference to the Microsoft Word 8.0 Object Library.



FIG. 31.23 Properties and methods of Word 97's `Bookmark` object displayed by Access 97's Object Browser.

One of the most effective methods of using Access 97 to create custom Word 97 documents is to define a set of named bookmarks and replace the bookmarked text with text from an Access Recordset object. In this case, a Word `Bookmark` object corresponds to the `Cells(x, y)` property of an Excel `Worksheet` object, where the `x, y` arguments specify a single cell. Figure 31.24 shows code in the Debug window for reading and writing bookmarked text in the Word file for the manuscript of this chapter (Uaw31rj.doc). `Selection` is a property of the `Application` object, not the `Document` object. Only one `Selection` can be active in an instance of Word, no matter how many `Documents` are open.

Oracle Ex. 1009, p. 1128
Oracle v. DeBellis
IPR2024-00221

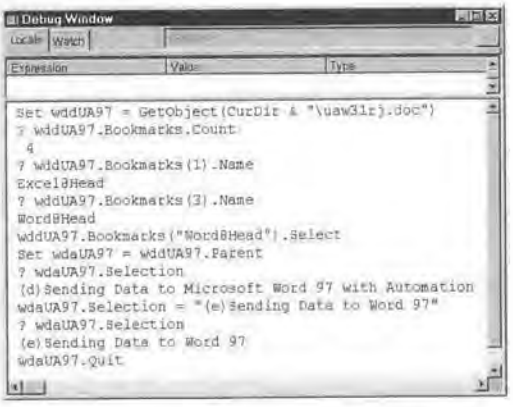


FIG. 31.24 Reading and replacing bookmarked text in a Word 97 document.

To create and save a new Word document, use the following generic code:

```
Private wdaName As Object 'Word Application object
Private wddName As Object 'Word Document object

Set wdaName = CreateObject("Word.Application")
wdaName.Documents.Add 'Empty document
Set wddName = wdaName.ActiveDocument

'... Code to manipulate document

wddName.SaveAs FileName:="filename.doc"
wdaName.Quit
Set wddName = Nothing
Set wdaName = Nothing
```

It isn't obvious how to add text to an empty document you create with the `Documents.Add` method. When you add an empty document, it contains a single character (a paragraph marker), which represents the `Content` property of the `Document`. The following code creates a `Range` object representing the entire document and then applies the `InsertBefore` method to add text to the empty document:

```
Private wdrName As Object 'Word Range object

Set wdrName = wddName.Content 'Range = 1 character
wdrName.InsertBefore "Text to insert"
```

Figure 31.25 is an example of using the `InsertBefore` method in the Debug window for adding text to a newly created document. The `Select` method of the `Range` object selects the entire `Range`. With the selection active, you can apply the `Add` method to the `Bookmarks` collection to add a persistent `Bookmark` object to contain the selection.

Programming a Word Document in an Access Object Frame. Most people who alter word processing documents want to view the modifications before final acceptance. Thus, the majority of Access applications that use Automation with Word 97 as the

Oracle Ex. 1009, p. 1129
Oracle v. DeBellis
IPR2024-00221

server are likely to involve Word documents contained in bound or unbound object frames. Chapter 22, "Using Access with Microsoft Word and Mail Merge," explains how to add a bound object frame containing an embedded Word document.

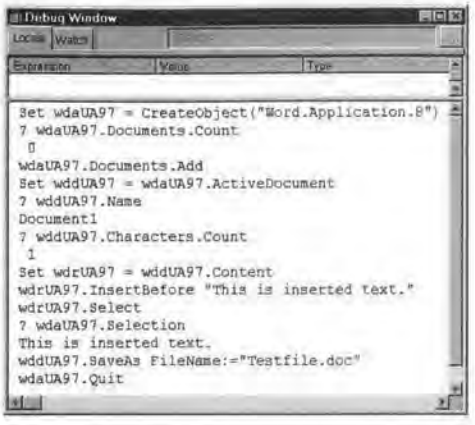


FIG. 31.25 Adding text to an empty Word 97 document with the `InsertBefore` method of the `Range` object.

◀◀ See "Embedding or Linking Word Documents in Access Tables," p. 778



The following generic code, with a reference to the Microsoft Word 8.0 Object Library, lets you apply Word 97 methods to the embedded document contained in a bound or unbound object frame:

```
Dim wdaName As Word.Application
Dim wddName As Word.Document

[Forms!FormName!]ControlName.Action = acOLEActivate
Set wdaName = [Forms!FormName!]ControlName.Object.Application
Set wddName = wdaName.ActiveDocument

'...Code to manipulate document

[Forms!FormName!]ControlName.Action = acOLEClose
Set wddName = Nothing
Set wdaName = Nothing
```

The *ControlName*`.Action = acOLEActivate` statement is required if the embedded object has not been activated when you execute the preceding code example, either in the Debug window or a VBA procedure. An embedded object in a frame must be activated before you can refer to its `Object.Application.`*PropertyName* property. Linked objects in object frames don't support Automation. You receive a "This object does not support OLE Automation" error message when you execute the `Set wdaName = uofName.Object.Application.WordBasic` for a linked file.

Oracle Ex. 1009, p. 1130
Oracle v. DeBellis
IPR2024-00221

Figure 31.26 shows an example of activating an embedded Word 97 object and then reading the value of bookmarked text using the `Selection` object. To execute the code shown in Figure 31.26, you must first insert an unbound Word 97 object created from a file with bookmarks that are named "Excel8Head" and "Word8Head" in a form named `frmWord`. You must also create a reference to the Microsoft Word 8.0 object library and declare the object variables wdaUA97 and wddUA97 in the declarations section of a module.

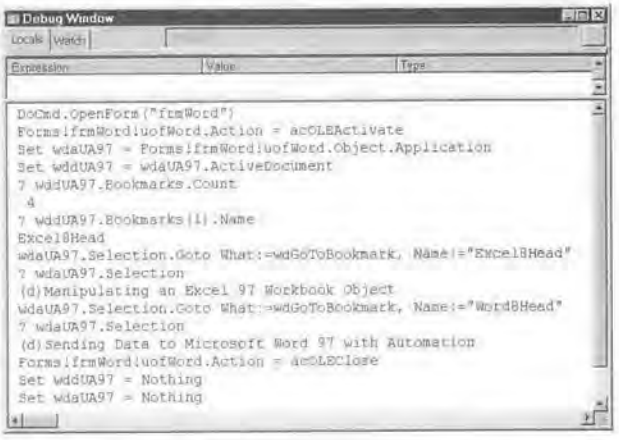


FIG. 31.26 Reading the text of a bookmark in an embedded OLE object.

By using Automation and code similar to that shown in Figure 31.26, you can create a document that consists of nothing but empty bookmarks. Then you can fill the bookmarks with text contained in Text or Memo fields of tables or from the `Value` property of control objects on forms. Using Automation provides a much more flexible method of creating specialized documents than using Access 95's Merge It feature.

Adding ActiveX Controls to Your Application

ActiveX Controls (OCXs) are a special class of programmable Automation objects that expose their own events in addition to their properties and methods. You embed the ActiveX Control in a form and program the Control with the same types of statements used with native Access controls, such as a text box or combo box. Some ActiveX Controls are databound, in which case you embed the Control in a form bound to an Access table or query.

ActiveX controls are similar to Visual Basic custom controls (VBXs) in concept and use. VBXs are 16-bit only, while ActiveX controls are 32-bit only. All of the Office 97 applications that support Forms 2.0 let you embed ActiveX controls on forms; Internet Explorer 3.0 also supports ActiveX controls. (Access 2.0 and 16-bit Visual Basic 4.0 use 16-bit OLE Controls, which now are obsolete.)

Oracle Ex. 1009, p. 1131
Oracle v. DeBellis
IPR2024-00221

The sections that follow describe the Calendar ActiveX Control supplied with Access 97; they explain how to register an OCX when necessary, and show you how to insert and program the Microsoft Calendar Control.

> **Note**
>
> Access 2.0 and Access 95 included the Data Outline control, a very large OLE Control that provided a form navigation system for Access applications. An ActiveX version of the Data Outline control is not included with Access 97.

Registering OLE Controls

ActiveX Controls must be registered before they appear in the list of Controls available for use with Access 97. Most commercial ActiveX Controls, including those supplied with Access 95 and Visual Basic 5.0, create the required Registry entries during the installation process. Figure 31.27 shows in RegEdit.exe the Registry entries for the Access 97 Calendar Control.

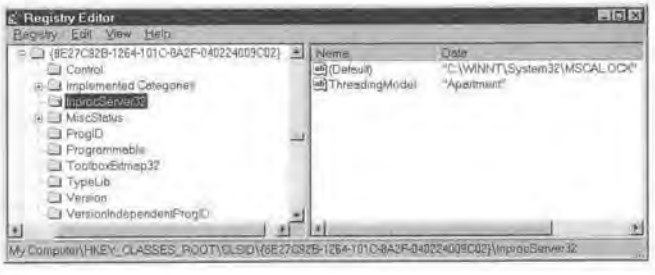


FIG. 31.27 The Registry hive for the Calendar Control displayed in RegEdit.exe.

Many shareware ActiveX Controls don't come with a Setup application, so you must register the OCXs yourself. If your Windows 95 or Windows NT Registry file becomes corrupted and you cannot solve the problem by restoring the Registry backup file, you must reregister your ActiveX Controls. (If the Registry file becomes irreparably corrupted, you probably will also need to reinstall all of your Windows applications and, perhaps, Windows 95 or Windows NT.)

To register a 32-bit ActiveX Control for Access 97 and for all other applications that use 32-bit OCXs, follow these steps:

1. Choose Tools, ActiveX Controls to open the ActiveX Controls dialog, and then click the Register button to open the Add ActiveX Control dialog.
2. Maneuver to the \Windows\System or the \Winnt\System32 folder, the standard locations for OLE controls, and select the .ocx file you want to register or reregister (see Figure 31.28).

Oracle Ex. 1009, p. 1132
Oracle v. DeBellis
IPR2024-00221

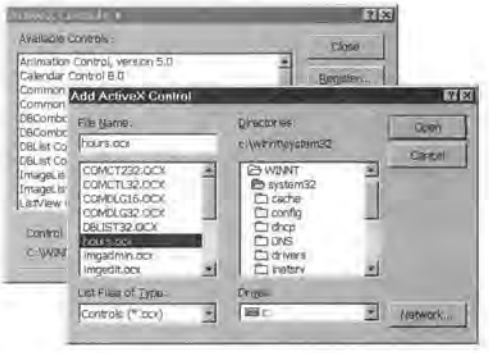


FIG. 31.28 Selecting an .ocx file to register.

3. Click the Open button to close the Add ActiveX Control dialog and register the Control. You can't register some .ocx files as Access ActiveX controls. If you attempt to register a special-purpose Control, you receive an error message stating that the control cannot be registered.

If you want to delete a custom control from your fixed disk, it is advisable to delete the Registry entry for the control before deleting the control. Select the name of the control in the Available Controls list of the ActiveX Controls dialog, and then click the Unregister button to remove the Registry entry for the control. Then delete the corresponding .ocx file.

Using the Calendar Control

The Microsoft Calendar Control was one of the three original OLE Controls introduced with Access 2.0. Mscal.ocx, included with Access 97, is an improved, 32-bit ActiveX version of the original Calendar Control. Mscal.ocx is a data-bound control; that is, you can bind the Control to a field of the Date/Time data type. The data-bound Calendar Control displays the date in calendar form.

The following sections describe how to add the Calendar Control to a form and how to program the Calendar Control.

Adding the Calendar Control to a Form. To add a Calendar Control in an unbound object frame of a form, follow these steps:

1. Create a new blank (unbound) form about 5 inches wide and 3 inches deep.
2. Choose Insert, ActiveX Control to display the Insert ActiveX Controls dialog.
3. Select the Calendar Control from the Select an ActiveX Control list (see Figure 31.29). Click OK to add the Calendar Control to the form.
4. Size the Calendar Control to about 3×2 inches, position the mouse pointer on the Control, and then click the right mouse button to open the popup menu for the Control.

Oracle Ex. 1009, p. 1133
Oracle v. DeBellis
IPR2024-00221

5. Choose Calendar Object from the floating menu with the left mouse button to open the object's submenu (see Figure 31.30). Choose Properties to open the properties sheet for the Control.

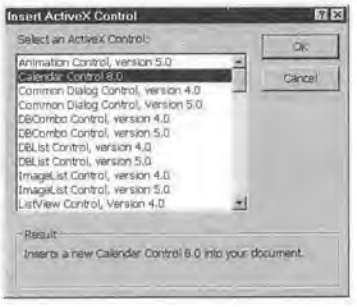


FIG. 31.29 Adding the Calendar Control with the Insert ActiveX Controls dialog.

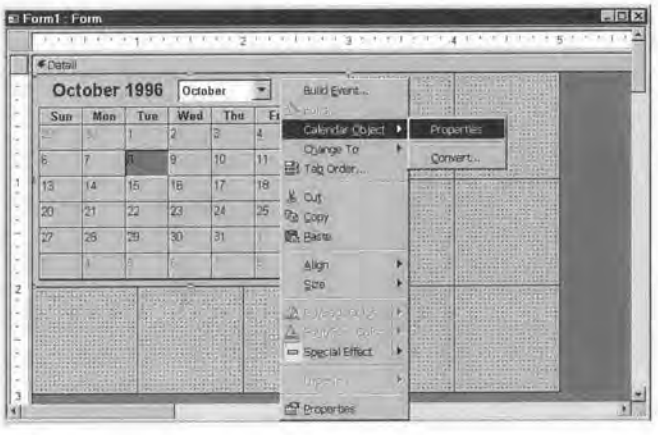


FIG. 31.30 The popup menus for an ActiveX Control.

6. The tabbed properties sheet opens to let you set a variety of General properties for the Calendar Control. Click the Font tab to open the Fonts properties page.

> **Note**
>
> Technically speaking, the properties dialog of an ActiveX Control is called a *property frame* or, more commonly, *properties sheet*, and the content of the property frame is stored in *property pages*. Entries for the property sheet and property page(s) appear as entries for the ActiveX control in the Registry.

7. With DayFont selected in the Property Name list, select Arial from the Font combo list, apply the Bold attribute, set the font size to 10 points, and click the Apply button (see Figure 31.31).

Oracle Ex. 1009, p. 1134
Oracle v. DeBellis
IPR2024-00221

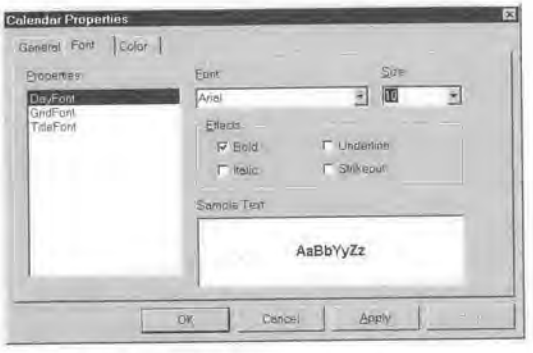


FIG. 31.31 The Fonts page of the Calendar Properties sheet.

8. Repeat the font selection process of the preceding step for the `GridFont` property (10-point Arial bold) and `TitleFont` property (14-point Arial bold).

9. Click the Colors tab to display the Colors properties page. You can change the color of the background (BackColor), DayFont, GridFont, TitleFont, and GridLines to one of the 16 standard colors or Windows system colors (see Figure 31.32).

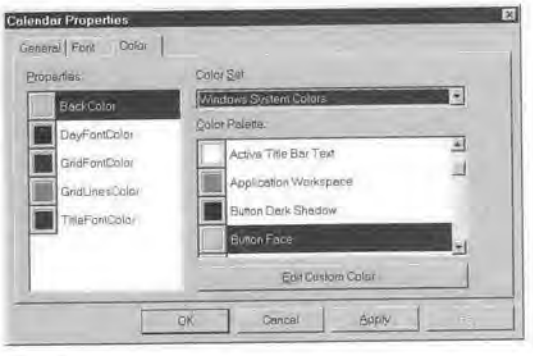


FIG. 31.32 The Color page of the Calendar Properties sheet.

10. Click OK to close the properties sheet. Double-click a blank area of the form to display the Form properties window. Then set the Scroll Bars property to Neither and the Record Selectors and Navigation Buttons properties to No. Change the name of the control from ActiveXCtrl1 to **ocxCalendar**. Your Calendar Control appears in Run mode, as shown in Figure 31.33.

11. To eliminate the month and year drop-down lists and change the appearance of the Control, return to Design view and open the properties sheet for ocxCalendar. In the General properties page, clear the Month/Year Selectors check box and select Sunken in the Grid Cell Effect list.

Oracle Ex. 1009, p. 1135
Oracle v. DeBellis
IPR2024-00221

12. Click OK to close the properties sheet. Change to Form view; the modified Calendar Control appears, as shown in Figure 31.34.



FIG. 31.33 The modified Calendar Control in Run mode with month and year drop-down lists.



FIG. 31.34 The modified Calendar Control in Run mode without the drop-down lists.

13. Save your form with a descriptive name, such as **frmCalendar**.

Programming the Calendar Control. Each ActiveX Control has its own collection of properties, methods, and events. Each of the ActiveX controls supplied with Access 97 and the ODE has an individual online help file to assist you in programming the control. You can open the help file by clicking the Help button of a Control's properties sheet, if present. The Help file for the Calendar Control is integrated with the Access 97 online help file. Another approach is to choose Start, Find, and type ***.hlp** in the Named text box of the Find: Files Named dialog. Then click the Find button. The help file usually is the file name for the control with an .hlp extension (see Figure 31.35).

Double-click the help file for the ActiveX Control to display the Help Topics dialog for the control (see Figure 31.36). Figure 31.37 shows the help topic for the `PreviousMonth` method for the Calendar Control.

Oracle Ex. 1009, p. 1136
Oracle v. DeBellis
IPR2024-00221

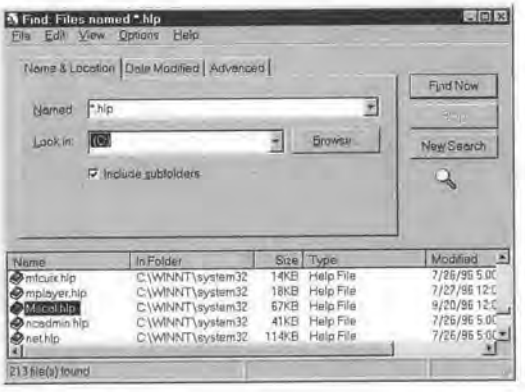


FIG. 31.35 Finding help files for OLE Controls in Windows NT 4.0.

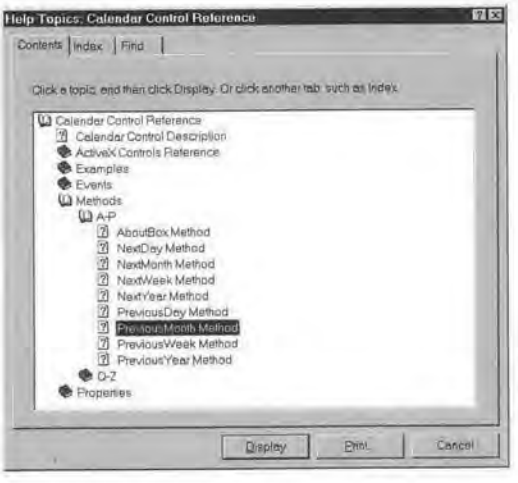


FIG. 31.36 The Help Topics dialog for the Calendar Control.

To program the Calendar Control and complete the design of the frmCalendar form, follow these steps:

1. Add six command buttons, stacked vertically to the right of the calendar. Assign **Next Week**, **Last Week**, **Next Month**, **Last Month**, **Next Year**, and **Last Year** as the value of the Caption property for the six buttons in sequence. Name the buttons **cmdNextWeek**, **cmdLastWeek**, and so forth.
2. Add an unbound text box under the calendar. Assign **lblDate** and **txtDate** as the values of the Name property of the label and text box, respectively (see Figure 31.38).

Oracle Ex. 1009, p. 1137
Oracle v. DeBellis
IPR2024-00221

3. Click the Calendar Control with the right mouse button and choose Build Event from the floating menu to open the class module for ocxCalendar. Select the `AfterUpdate` event from the events drop-down list to create the **`Private Sub`** `ocxCalendar_AfterUpdate()...` **`End Sub`** stub. Note that the events exposed by the Calendar Control, such as `AfterUpdate`, `NewMonth`, and `NewYear`, appear in the list of events for the ocxCalendar control.

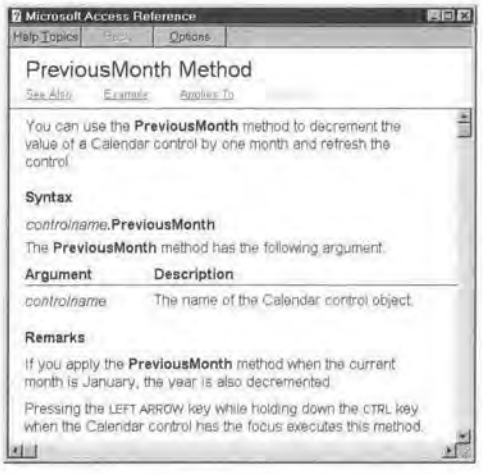


FIG. 31.37 The online help topic for the `PreviousMonth` method of the Calendar Control.



FIG. 31.38 The final design of the Calendar Control form.

4. Type the following code (see Figure 31.39) for the `AfterUpdate` event to display the date when the form opens and when the date is changed:

```
Private Sub ocxCalendar_AfterUpdate ()
   'Purpose: Update the text boxes
```

Oracle Ex. 1009, p. 1138
Oracle v. DeBellis
IPR2024-00221

```
    lblDate.Caption = Format(ocxCalendar.Object.Value, _
       "mm/dd/yy")
    txtDate.Value = Format(ocxCalendar.Object.Value, "dddddd")
End Sub
```

Note

Access 97 inserts ActiveX Controls in bound or unbound object frames, so you must specify the `Object` property of the frame to gain access to the properties and methods of the control.

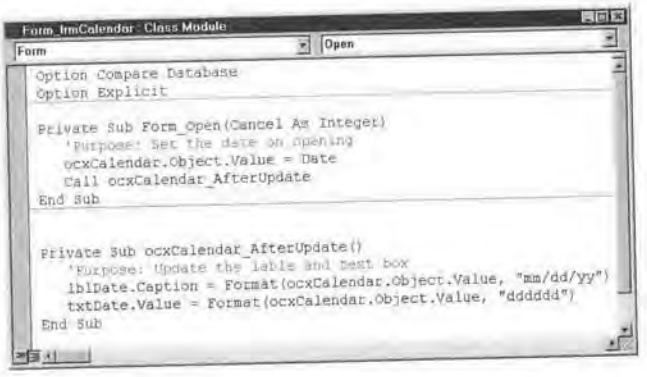


FIG. 31.39 Entering the code for the `AfterUpdate` event.

5. Enter the following code to set the date explicitly when opening the form:

```
Private Sub Form_Open(Cancel As Integer)
   'Purpose: Set the date on opening
   ocxCalendar.Object.Value = Date 'Explicitly set the date
   Call ocxCalendar_AfterUpdate
End Sub
```

6. Mscal.ocx has methods for changing the week, month, and year (refer to Figure 31.37), so you can add the following code to apply the appropriate method to the Calendar Control with each `Click` event for each of the command buttons:

```
Sub cmd{Next¦Previous}{Week¦Month¦Year}_Click ()
   'Purpose: Use the Next or Previous period method
   'to increment or decrement the period
   ocxCalendar.Object.{Next¦Previous}{Week¦Month¦Year}
End Sub
```

7. Complete the design of the form by setting the value of the `MinMaxButtons` property to `None`.

8. Click the Form View button of the toolbar to open your Calendar Control form (see Figure 31.40). Verify your event-handling procedures by testing the action of each of the six buttons.

Oracle Ex. 1009, p. 1139
Oracle v. DeBellis
IPR2024-00221

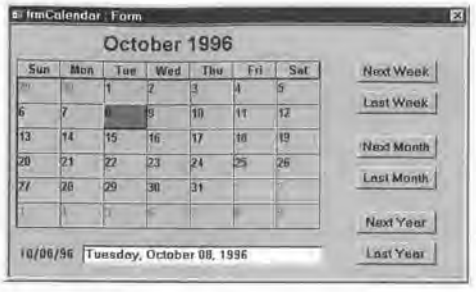


FIG. 31.40 The finished Calendar Control in Form view.

Using Access 97 as an Automation Server

Access 97 exposes all of its application objects to other 32-bit, automation-enabled applications through the `Access.Application.8` object. To use Access as an automation server, you create in a module a reference to the Microsoft Access 97 type library, Msaccess.tlb—which is located in the Access folder, not in \Windows\System. Any member of the Office 97 suite and 32-bit Visual Basic 4+ can manipulate Access 97 `Application` objects, properties, and methods.

The extent to which developers will use Access 95 application objects instead of the Jet 3.5 database engine remains to be seen. Using the Jet 3.5 data access object to manipulate databases is much faster and consumes far fewer resources than launching an instance of Access to perform ordinary database-related operations.

You use the `CreateObject` function to open a minimized instance of Access and then open a database with the `OpenCurrentDatabase "DatabasePathName"` method. Alternatively, you can use the `GetObject("path\filename.mdb")` function. From this point on, you program the objects with Access VBA code in the other application. You even can invoke Access VBA's `DoCmd` object from the Automation client with `acaName.DoCmd.Action Argument(s)` statements.

As with using Access to manipulate objects of other applications described earlier in this chapter, the Debug window is the best way to gain familiarity with the Access Application object. To experiment with Access 97's Application object in Excel 97, follow these steps:

1. Close Access if it is open, and launch Excel 97.
2. Choose Tools, Macro, Visual Basic Editor to open an empty Module1 class module.
3. Choose Tools, References to open Excel's References dialog. In the Available References list, mark the check box for Microsoft Access 95 (see Figure 31.41). Mark the Microsoft DAO 3.5 check box if you want to use database intrinsic constants. Click OK to close the dialog.

Oracle Ex. 1009, p. 1140
Oracle v. DeBellis
IPR2024-00221

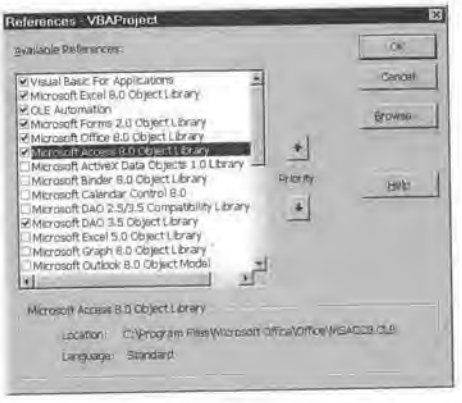


FIG. 31.41 Adding a reference to the Microsoft Access 8.0 Object Library to Excel 97.

4. Press F2 or choose View, Object Browser to open Excel's Object Browser. Select Access from the Project/Library list to display the objects exposed by the Microsoft Access 97 Object library (see Figure 31.42). Close Object Browser when you've finished exploring the Access `Application` objects.

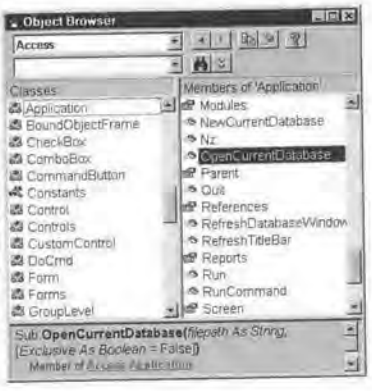


FIG. 31.42 Some of the Access Application objects displayed in Excel 97's Object Browser.

5. Add the following object variable declarations to the module:

```
Private acaNwind As Access.Application
Private dbNwind As Database
Private rstNwind As Recordset
```

6. Press Ctrl+G or choose View, Immediate Window to open Excel's Immediate window—the equivalent of Access 97's Debug window.

7. Type **Set acaNwind = CreateObject("Access.Application.8")** in the Immediate window. When you press Enter, the command eventually launches an invisible instance of Access.

Oracle Ex. 1009, p. 1141
Oracle v. DeBellis
IPR2024-00221

8. When <Ready> appears in the status line, type **acaNwind.OpenCurrentDatabase "northwind.mdb"**. (Add the path if you don't have a copy of Northwind.mdb in the My Documents folder.) Pressing Enter opens Northwind.mdb.

9. Type **Set dbNwind = acaNwind.CurrentDB( )**. After executing this statement, most of the commands to manipulate database objects are identical to those of Access VBA.

10. Type **Set rstNwind = dbNwind.OpenRecordset("Customers")** to open a `Recordset` object over the Customers table. Apply typical methods and check some property values of the `Recordset` object, as shown in Figure 31.43.

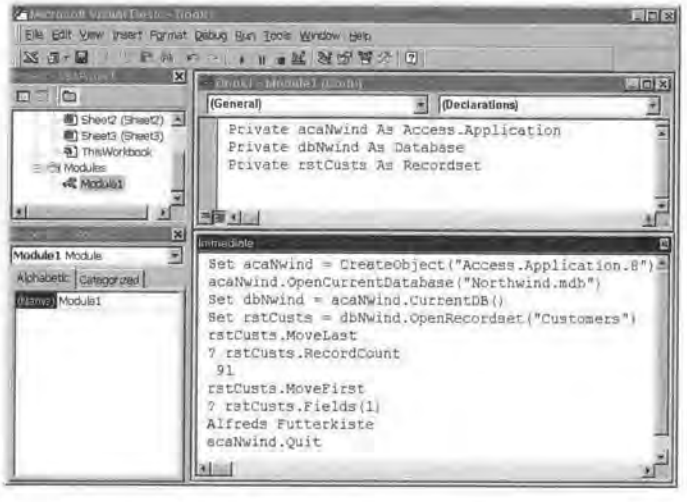


FIG. 31.43 Creating and manipulating an Access 97 `Recordset` object with Excel VBA.

11. When you finish experimenting with Access Application objects, type **acaNwind.Quit** to close the instance of Access.

Experimenting with Access as an Automation server demonstrates that Visual Basic for Applications truly is the *lingua franca* of application programming (also called *macro*) languages. Although the hierarchy of the objects exposed to VBA varies with server applications, the programming language and methodology is independent of the client and server applications. Now that Microsoft Word has adopted Word VBA, Microsoft's Bill Gates has finally achieved his 1989 vision of a "common macro language" for all Microsoft productivity applications.

Oracle Ex. 1009, p. 1142
Oracle v. DeBellis
IPR2024-00221

Chapter 32

Using the Office 97 Developer Edition

The Office 97 Developer Edition (ODE) differs greatly from the Access Developer's Toolkit (ADT) of Access 95 and Access 2.0. Earlier versions of Access used a separate run-time executable file, MSARN??0.EXE; like Access 95, Access 97 uses the retail version of Msaccess.exe set to run-time mode for distributing applications. Even if you don't intend to distribute your applications to others, you might want to acquire the ODE because of the additional ActiveX Controls and printed documentation the ODE provides.

This chapter describes what's in the ODE, how to use the Setup Wizard to create run-time distribution disks for your Access 97 applications, and how to use the ODE's API Viewer to add 32-bit Windows (Win32) API function prototype declarations to your Access VBA code.

Examining the Content of the Office 97 Developer Edition

The new ODE for Office 97 is directed primarily toward the Access developer and Office power-user communities. The ODE contains the following components:

- A royalty-free license to distribute Msaccess.exe for run-time use, the run-time interpreter for VBA 5.0 (Vbrun500.dll), the run-time version of Microsoft Graph 8.0 (Graph8.exe and Graph8rt.srg), and the other distributable components of Access 97, such as the Jet 3.5 database engine, ActiveX Controls used by your application, ODBC files, and ISAM drivers.
- An improved version of the Setup Wizard you use to create images of the distribution disks for your application. The Setup Wizard also writes the Setup.inf and other data files required by Setup.exe.
- A set of three printed manuals: *Microsoft Office 97 Programmers Guide*, *Building Applications with Microsoft Access 97*, and *Microsoft Office 97 Object Model Guide*. The *Programmers Guide* provides an introduction to

Oracle Ex. 1009, p. 1143
Oracle v. DeBellis
IPR2024-00221

VBA 5.0 programming for all of the members of Office 97. *Building Applications* explains how to design database application with Access 97, other Office 97 members, and Jet 3.5. The *Object Model Guide* supplies object model diagrams and lists the properties and methods of application-specific objects.

- The 32-bit Animation, Common Dialog, FTP, Gopher, HTTP, Imagelist, Listview, Progress Bar, Rich Text, Slider, Status Bar, Tabstrip, Toolbar, Treeview, Up/Down, and Winsock ActiveX controls.
- The Replication Manager (Replman.exe) to automate Briefcase replication operations between multiple sites. The new Replication Manager lets you replicate Access databases over the Internet and supports partial replication of databases.
- The Microsoft Help Compiler version 4.0 for Windows (Hcrtf.exe), the graphic front-end for the help compiler (Hcw.exe), and help files for the compiler.
- A Windows API Viewer application, Apilod.exe, which provides declarations in VBA 5.0 format for all of the Win32 API functions. The API Viewer aids in converting Access 2.0 16-bit Windows API function prototype declarations to the Win32 API.
- Components necessary to manage multi-developer Access 97 projects with Visual SourceSafe 5.0. You must acquire the Visual SourceSafe 5.0 application to use this feature. (Visual SourceSafe 5.0 is included in the Enterprise Edition of Visual Basic 5.0).

Except for the Help Compiler and API Viewer, all of the preceding components are new or have been significantly upgraded for Access 97 and other Office 97 applications.

Note

The standard version of the ODE includes a copy of Office 97. If you already have the Professional Version of Office 97 or a stand-alone copy of Access 97, you can purchase the ODE as an add-on product.

Differences Between Run-Time and Retail Access

The behavior of your 32-bit Access applications differs when you execute the applications for use with the run-time, rather than the full (retail), version of Access 97. Following are the principal differences that distinguish run-time execution:

- All design-related menu choices and corresponding object views are removed. Your run-time applications should use menu macros to create the menubars and menu items your application requires.
- Built-in toolbars aren't supported, but you can create custom toolbars for your application. (Custom toolbars are stored in the System.mdw file you distribute with your application.)

Oracle Ex. 1009, p. 1144
Oracle v. DeBellis
IPR2024-00221

- The Database window is not visible, and macro and module windows are hidden. You specify the form to open by selecting the form in the Startup dialog opened by choosing Tools, Startup.
- Special-purpose keys and combinations are disabled. Shift is disabled during the database opening process. Ctrl+Break is disabled so that users cannot halt Access VBA code or macro operation. (Ctrl+Break remains active during query execution so that users can halt a "run-away" query that returns very large numbers of records.)
- Pressing F1 for online help results in an error message if you don't supply a custom online help file for your application and assign the file name to the Help File property of every visible object of your application. (You also must include a valid Help Context ID value.) Supplying even a very simple help file is a better practice than allowing an error to occur when the user presses F1. (You are not allowed to distribute the Access online help files.)
- All errors that occur in the execution of macros, and untrapped errors that occur while executing Access VBA code, result in an instantaneous exit of Access. The user receives no warning of the impending disappearance of your application. You cannot trap macro errors, so you should use Access VBA code with error trapping rather than macros for all event-handling operations. There is no assurance the future versions of Access will support macros.
- If you use separate data and application .mdb files, you need to provide Access VBA code that tests to determine whether the *Data*.mdb file is on the expected drive and in the specified folder. If not, a special form and another routine is required to specify the location of the *Data*.mdb file.

No significant change in the preceding list has occurred with Access 97. Thus, the Access 95 run-time applications you convert to Access 97 are likely to behave identically. Chapter 33, "Migrating Access 2.0 and Access 95 Applications to Access 97," describes the changes you need to make to execute your applications with run-time Access 97. These changes also are necessary for the applications to execute with the retail version of Access 97.

Tip

You can emulate run-time operation by creating a desktop shortcut that includes the path and file name with the `/Runtime` command-line switch for Msaccess.exe in the Target text box of the Shortcut page of the Shortcut Name Properties sheet. As an example, type

C:\Program Files\Microsoft Office\Office\Msaccess.exe C:\Program Files\Microsoft Office\Office\Samples\Nwind.mdb /Runtime

in the Target text box to specify the .mdb file used in the following example. It's a good practice to specify the current directory in the Start In text box, **C:\Program Files\Microsoft Office\Office** for this example.

Oracle Ex. 1009, p. 1145
Oracle v. DeBellis
IPR2024-00221

Creating Distribution Disks with the Setup Wizard

Microsoft Corporation's standard Setup program changed with the introduction of the Windows 95 round of Microsoft productivity application upgrades. The Setup application for users of your Access 97 run-time applications uses the standard Microsoft Office 97 Setup program, often called *Acme Setup*. The Access 97 Setup Wizard (Wzstp80.mde, an encrypted library) automatically creates the custom .stf and .inf files required by the Acme Setup program.

Preparing Sample Files to Use the Setup Wizard

The description of using the Setup Wizard that follows assumes that you have created or appropriated an icon file for your application and have an online help and help contents files. The following steps prepare the files necessary to create run-time disk images:

1. The example uses a split version of the Northwind.mdb file (Nwind.mdb as the application file and Nwind_be.mdb as the data file). Compact Northwind.mdb into a copy named Nwind.mdb, open Nwind.mdb, and use the Database Splitter Add-In to move the tables to Nwind_be.mdb in your \Program Files\Microsoft Office\Office\Samples folder, the default folder for this example.



◀◀ See "Splitting Databases for File Sharing," p. 873

Caution

Make sure that you haven't marked the Break on All Errors check box in the Error Trapping Group of the Advanced Page of the Options properties sheet. If Break on All Errors is marked, you receive a `Runtime Error 3025: Item not found in this collection` message when running the Setup Wizard.

2. Copy any icon (*Iconname*.ico) file on your disk to the default folder. (Choose Start, Find, Files or Folders and enter *.ico in the Named text box, then Find Now to locate an icon file to copy.) Rename the icon file copy to **Nwind.ico**.
3. If you don't have your own help files, for this example you can use the Nwind80.hlp and Nwind80.cnt files provided with the Northwind.mdb sample database.
4. Copy \Program Files\Microsoft Office\Office\System.mdw to the default folder. You cannot create a distribution copy of the System.mdw file that Access automatically opens when launched.

Creating Distribution Disk Images

To create images of the distribution disks for the Northwind Traders (Runtime) application on your fixed disk (or a server drive), follow these steps:

Oracle Ex. 1009, p. 1146
Oracle v. DeBellis
IPR2024-00221

Tip

Make sure you have at least 40M of free disk space on the drive that you use to store the disk images before you start creating distribution disks. The following example requires about 35M to store the compressed disk images and component files.

1. Open Nwind.mdb and choose Tools, Startup to open the Startup dialog. Type your application's name, such as Northwind (Runtime), in the Application Title text box; click the Builder button and select Nwind.ico in the Icon Browser dialog; and then select Main Switchboard from the Display Form drop-down list (see Figure 32.1). Accept the remaining defaults and click OK to close the dialog.

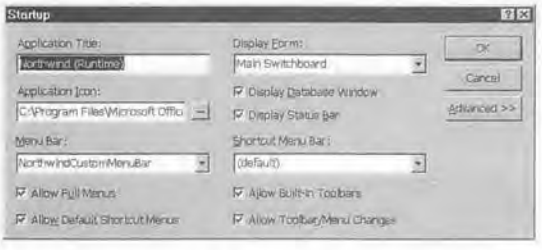


FIG. 32.1 Setting the startup parameters for Nwind.mdb.

2. Open the Wzstp80.mde database in your \Program Files\Microsoft Office\ODE Tools\Setup Wizard folder. (If Access isn't running, launch the wizard by choosing Start, Programs, Microsoft ODE Tools, Setup Wizard.) The Setup Wizard displays the opening dialog shown in Figure 32.2. Accept the default Create a New Set of Setup Options choice and click Next to display the second wizard dialog.

Next >

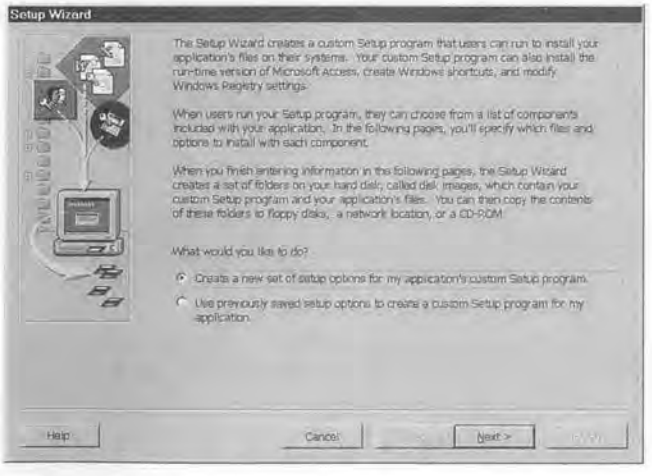


FIG. 32.2 The opening dialog of the Setup Wizard.

Oracle Ex. 1009, p. 1147
Oracle v. DeBellis
IPR2024-00221

3. Click the Add button to open the Select Files dialog. Press the Ctrl key and select Nwind.ico, Nwind.mdb, Nwind_be.mdb, Nwind80.cnt, Nwind80.hlp, and System.mdw (see Figure 32.3). Click Add to add the files to List of Files list box of the Setup Wizard's dialog.



FIG. 32.3 Selecting the required application files in the \Program Files\Microsoft Office\Office\Samples folder.

> **Note**
>
> For a multiuser application, install Nwind_be.mdb and System.mdw on the file server and do not include either of these two files in the List of Files. Create a second Setup program to install the shared database and workgroup files on the server.

4. Select Nwind.mdb in the List of Files list box and mark the Set As Application's Main File check box (see Figure 32.4). Accept the default `$(AppPath)` for the Destination Folder, Older for Overwrite Existing File, and accept Application as the Component Name for the file.

5. Select System.mdw and mark the Set as Workgroup File check box, as shown in Figure 32.5. If you include the workgroup file, this is an important step because marking the Set As Workgroup File check box causes setup to make the appropriate Registry entries for System.mdw.

> **Note**
>
> For multiuser applications, you must include the Workgroup Administrator application (Wrkgadm.exe from \Program Files\Microsoft Office\Office) so that the user can specify the network location of the workgroup file before launching your application. You add the Workgroup Administrator application, if required, in step 10 of this procedure.

Oracle Ex. 1009, p. 1148
Oracle v. DeBellis
IPR2024-00221

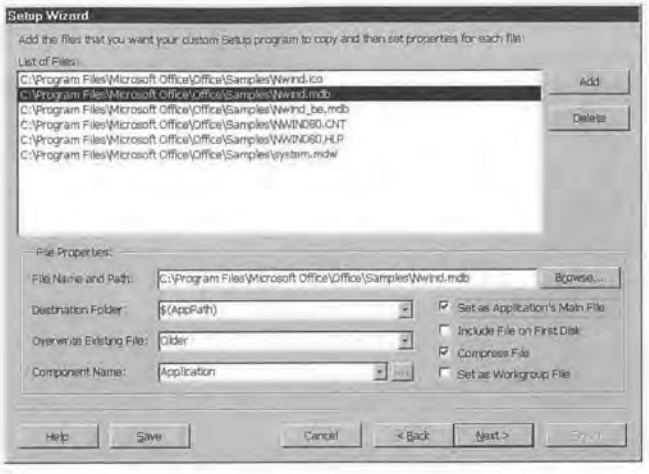


FIG. 32.4 Adding the application files to the List of Files list box and setting the application's main file.

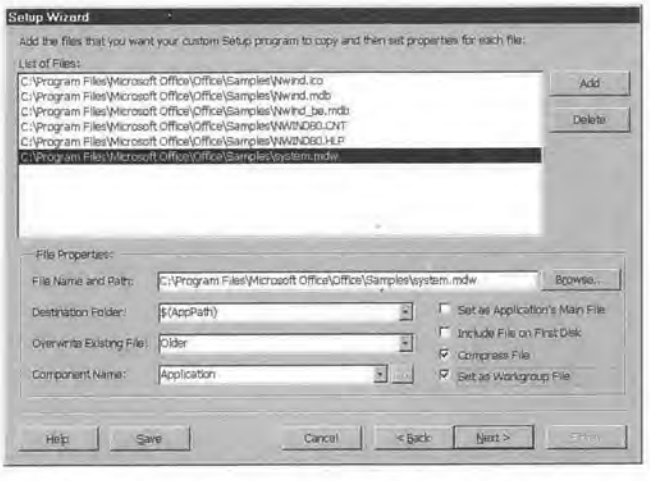


FIG. 32.5 Specifying System.mdw as the workgroup file for a self-contained application.



6. With System.mdw selected, click the Builder button (to the right of the Component Name drop-down list) to open the Components Builder dialog, then click the Add button. Type Workgroup File in the Name text box and click the Close button to add Workgroup File to the Components list. Optionally, you can add other component names, such as Database and Help Files, before you close the Components Builder (see Figure 32.6). You can change the order of the components with the up and down arrows to the left of the List of Components.

Oracle Ex. 1009, p. 1149
Oracle v. DeBellis
IPR2024-00221

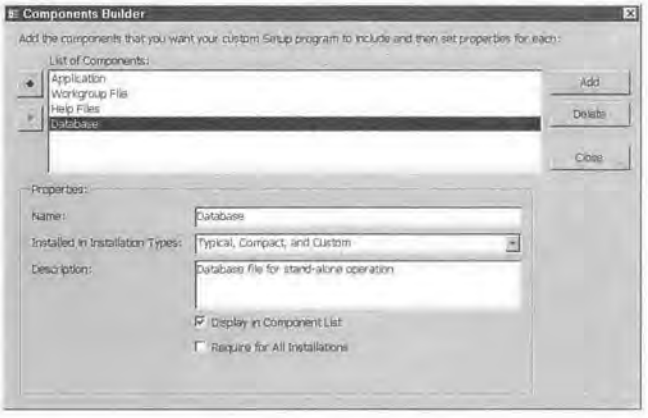


FIG. 32.6 Adding new component categories to the List of Components in the Component Builder dialog.

7. With System.mdw selected, open the Component Name drop-down list and select Workgroup File (see Figure 32.7). If you added additional components in the preceding step, select each file and assign the appropriate component name. Click Next to continue.

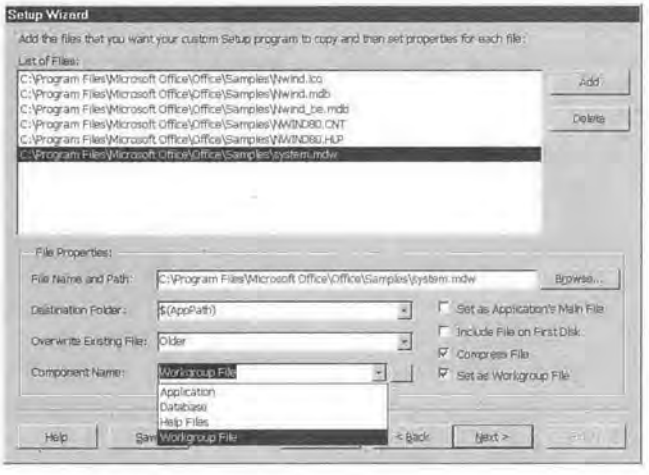


FIG. 32.7 Assigning a component name to the workgroup file.

8. The wizard lets you create custom shortcuts and shortcuts to run, compact, and repair your application. To provide a shortcut to launch Nwind.mdb, click the Database Shortcut Properties tab and, with the Open option selected, type **Northwind Runtime** in the Description text box. Mark the Run-Time check box, accept the remaining defaults, and click the Add button. Click Yes when the message box asks if you want to include the Access run-time files on your distribution disks. Repeat this process for Compact Northwind (select the Compact option button) and Repair Northwind (select the Repair and Compact option button).

Oracle Ex. 1009, p. 1150
Oracle v. DeBellis
IPR2024-00221

Figure 31.8 shows the three basic shortcuts added with the Open shortcut selected. You also can add a shortcut for your help file by selecting Nwind.hlp in the File to Open drop-down list and adding a **Northwind Help** description. Click Next to continue.

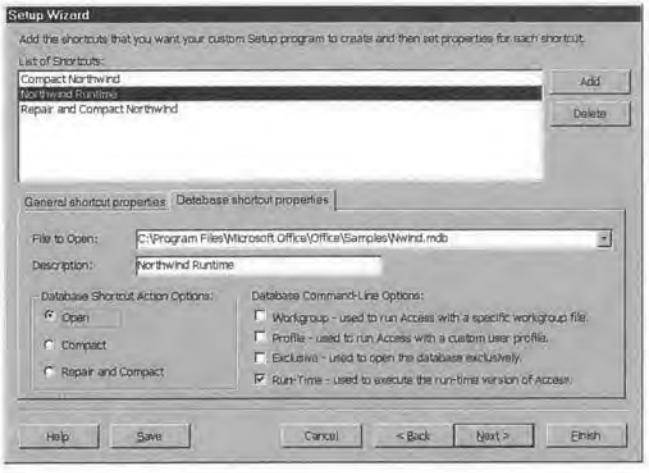


FIG. 32.8 Creating shortcuts for opening, compacting, and repairing the Nwind.mdb file.

9. The Setup Wizard automatically creates all of the Registry keys and values needed to run your application. You can add custom Registry keys, if needed, in the wizard's dialog shown in Figure 32.9. For this example, click Next to continue.

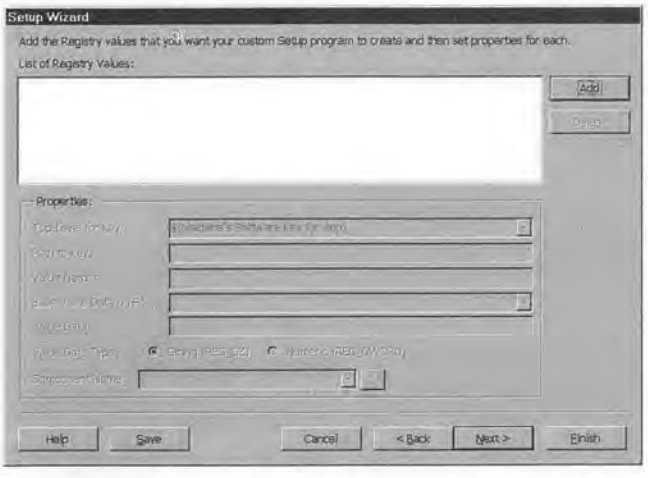


FIG. 32.9 Adding custom Registry keys, if required, for your run-time application.

10. Click the entries in the list box to select additional components to include on your distribution disks or in your network setup directory. An X indicates selected components (see Figure 32.10). If you answered Yes to the message described in step 8,

Oracle Ex. 1009, p. 1151
Oracle v. DeBellis
IPR2024-00221

the wizard selects Microsoft Access RunTime for you; if not, make sure to select this component. Also make sure to select Microsoft Graph 8 RunTime, if your application includes graphs or charts. Click Next to continue.

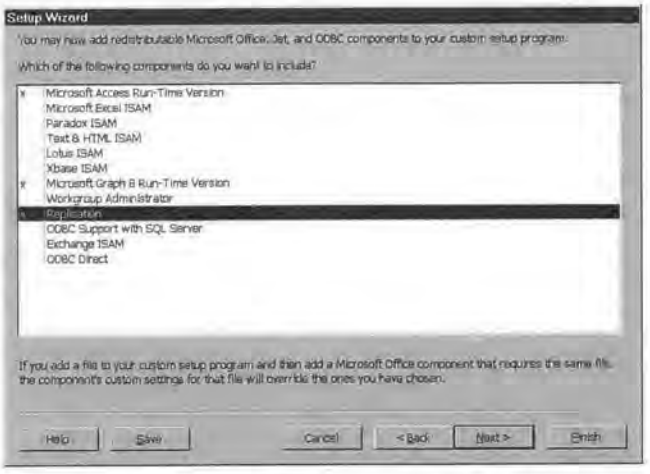


FIG. 32.10 Adding required components to the distribution files.

> **Note**
>
> You must include the Workgroup Administrator application with multiuser applications so that users can connect to the workgroup file on the workgroup server.

11. Verify that the components required for your application are installed in all three installation types, Compact, Typical, and Custom (see Figure 32.11). Mark the Require for All Installations check box for needed files. (Require for All Installations prevents users from failing to install required files when choosing Custom setup.) If you supply Replication Manager, you can select Custom from the Installed in Installation Types drop-down list so that Replication Manager only is installed when needed. Click Next to continue.

12. Type the name and version number of your application, your firm name, and the drive and name of the folder that corresponds to $(AppPath). The wizard provides default values based on previously entered information. Provision is made for a conventional DOS folder name, if your application is installed on a PC that doesn't support long file names, such as Windows 3.1+ and versions of Windows NT earlier than 3.51 (see Figure 32.12). Click Next to continue.

13. If you want to run an executable file when the setup operation completes, select the .exe file in the Run the Following File drop-down list. The executable file and any supporting .dll files must have been included in the file list created in preceding step 3, unless the file is known to be on the user's computer. You can include a Readme.wri file and type **Wordpad.exe $(AppPath)\Readme.wri** in the Enter

Oracle Ex. 1009, p. 1152
Oracle v. DeBellis
IPR2024-00221

or Edit the Command Line text box to display the file on completion of setup (see Figure 32.13). For a Readme file, be sure to mark the Allow Setup to Complete check box so Setup announces that it has completed successfully before your Readme file appears. Click Next to continue.

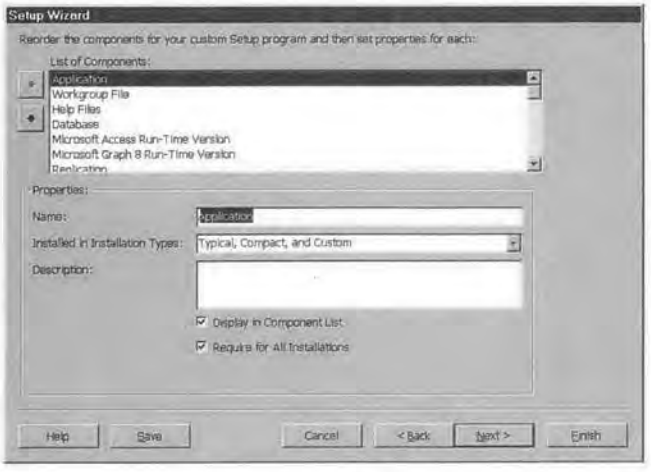


FIG. 32.11 Setting the property values of installation components.

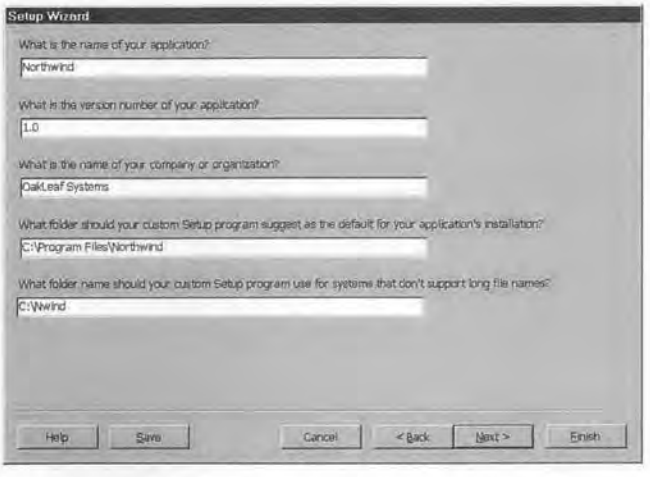


FIG. 32.12 Setting the property values for your run-time application.

14. Enter the path and folder name of the directory in which to store the disk images, each of which is stored in a ...\Disks\Disk1...\Disk*n* folder (see Figure 32.14). Alternatively, you can elect to create a single network setup folder in a server share. Compressed network setup files save space on the server, but installation takes slightly longer. Storing a local copy of compressed files makes re-creating your disk images or server files quicker. Click Finish to create your disk or server file images.

Oracle Ex. 1009, p. 1153
Oracle v. DeBellis
IPR2024-00221

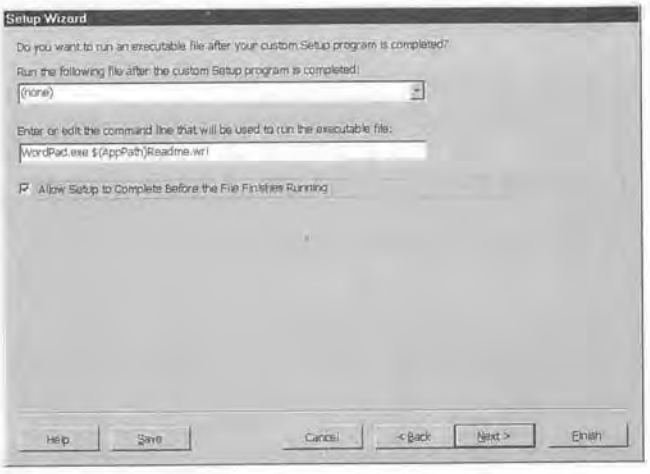


FIG. 32.13 Specifying an executable file to run on completion of the setup operation.

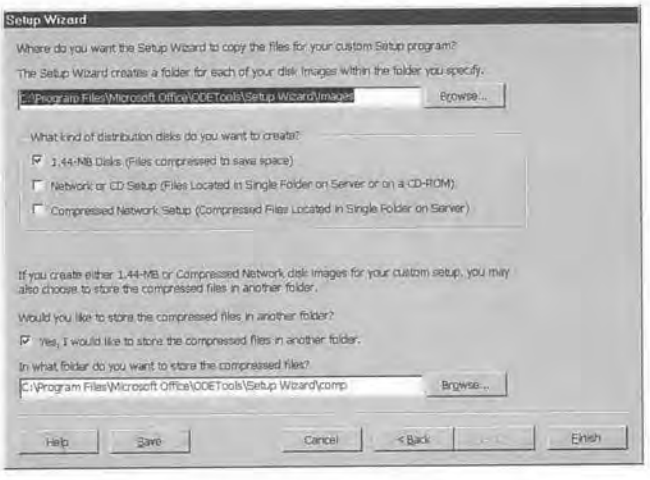


FIG. 32.14 Specifying the location of network or disk image distribution files and optional compressed files.

15. The wizard asks whether you want to save your setup data as a template. Click Yes to open the Save Template dialog (see Figure 32.15). Give your template an appropriate name, such as **Nwind.mdt**, and click the Save button to close the dialog.

16. The wizard starts creating the compressed files and disk images and provides a status dialog as shown in Figure 32.16. This is a good time to have a cup of coffee (or lunch), because the process may take five to 30 minutes, depending on the size of your application files, the component options you select, and the speed of your PC and disk drive. When the process completes, you receive the message shown in Figure 32.17.

Oracle Ex. 1009, p. 1154
Oracle v. DeBellis
IPR2024-00221



FIG. 32.15 Assigning a name to and saving a template for your setup data.



FIG. 32.16 The Setup Wizard keeps you informed of the progress of setup file generation.



FIG. 32.17 Success at last; the wizard notifies you of the completion of the process.

The setup files that the wizard creates for your application result in an installation process that is very similar to that for Access 97 and other members of the Office 97 suite. You can quickly check your Setup application by executing Setup.exe from your ...\Images\Disks\Disk1 folder and displaying the first few windows. Figure 32.18 shows the opening Setup window for the example created in the preceding steps. To fully validate your Setup application, however, you need to run Setup.exe from the disks or network file server on a workstation without either the retail or run-time version of Access 97 installed.

Using the 32-Bit Windows API Viewer

The Win32 API Viewer, which also is included with Visual Basic 4.0 and 5.0, is intended as an aid to converting your existing 16-bit Windows 3.1+ (Win16) API calls to Win32 format. To add Win32 function prototype declarations to your Access VBA code, follow these steps:

1. Launch the API Viewer, Apilod.exe, from your \Program Files\Microsoft Office\ODE Tools\Win API Viewer folder. You also can run the API Viewer by choosing Start, Programs, ODE Tools, Win32 API Viewer.

Oracle Ex. 1009, p. 1155
Oracle v. DeBellis
IPR2024-00221

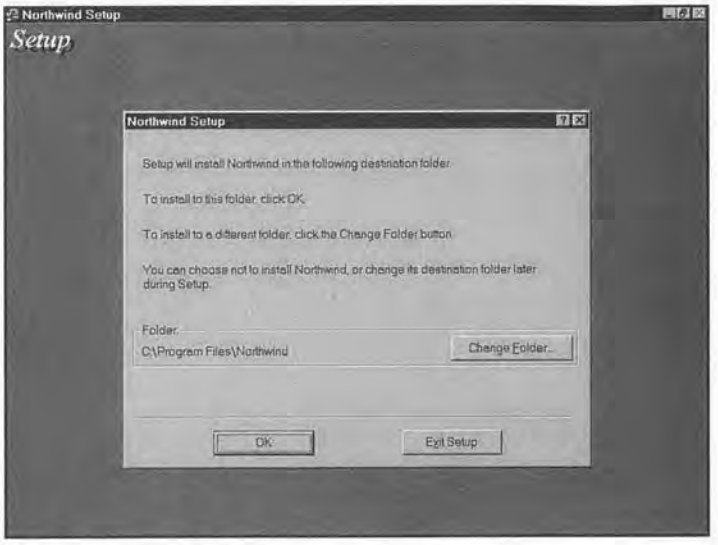


FIG. 32.18 The first window of the example Northwind setup program.

2. Choose File, Load Database File to open the Select a Jet Database dialog and select Win32api.mdb (see Figure 32.19). Click the Open button to load the database file and close the dialog.

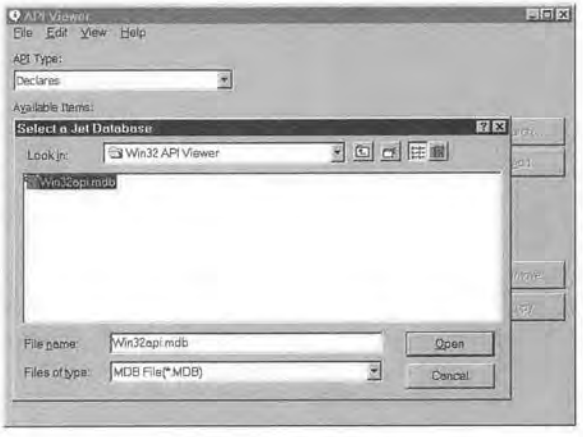


FIG. 32.19 Loading the Win32api.mdb file into the API Viewer.

3. With Declares selected in the API Type drop-down list, click the Available Items list box to give it the focus. Type the initial letter(s) of the Windows API function you want to add to your code. The corresponding API function entries appear.

Oracle Ex. 1009, p. 1156
Oracle v. DeBellis
IPR2024-00221

4. Select the API function you want to include in the Declarations section of your Access VBA code and click the Add button to add the entry to the Selected Items list box. Repeat this process for as many API functions as you need (see Figure 32.20).



FIG. 32.20 Selecting the function prototype declarations to add to your Access VBA code.

> **Note**
>
> The function prototype declarations of Win32api.mdb use C-style naming conventions (variable prefixes), instead of the Leszynski Naming Conventions (LNC) used in this book and described in Appendix B, "Naming Conventions for Access Objects and Variables." As an example, `lpHelpFile` (long pointer to a string) corresponds to `strHelpFile` using LNC. Win32api.mdb uses both `w` (word, usually 16 bits) and `dw` (double-word, 32 bits) as prefixes for the LNC `lng` (**`Long`**) data type prefix.

5. Click the Copy button to copy the function prototype declarations to the Clipboard.
6. Launch Access, if necessary, and open an existing or a new module.
7. Position the cursor at the line below the **`Option`** `Explicit` line in the Declarations section of the module and press Ctrl+V to paste the copied declarations.
8. Reformat the function prototype declarations using the line continuation pair (space plus underscore), as shown in Figure 32.21. Reformatting the declarations makes your code easier to read.

You also can use the API Viewer to add Win32 structures (user-defined data types) and constant declarations to your Access VBA code. The API Viewer is capable of using either specially formatted .txt or .mdb files, and can convert .txt files into .mdb files for faster access to a specific function.

Oracle Ex. 1009, p. 1157
Oracle v. DeBellis
IPR2024-00221

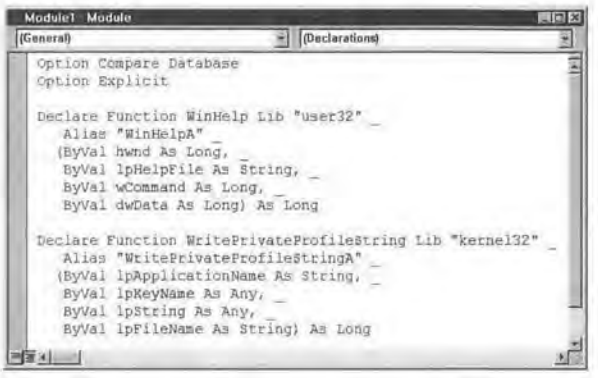


FIG. 32.21 The function prototype declarations pasted into a module and reformatted for readability.

Oracle Ex. 1009, p. 1158
Oracle v. DeBellis
IPR2024-00221

Chapter 33

Migrating Access 2.0 and Access 95 Applications to Access 97

Each version of Access—1.0, 1.1, 2.0, 95, and 97—has a different database file structure at the binary (byte) level. The differences between .MDB files created with versions 1.0 and 1.1 were relatively minor; thus, you could use the Compact feature of Access to convert version 1.0 .MDBs to version 1.1, or vice versa. The file formats of later versions of Jet databases are sufficiently different to require one-way conversion during the upgrade process.

The Convert Database process is not reversible; once you've converted a database to the Access 97 (version 8.0) .mdb structure of Jet 3.5, you can't convert it back to version 7.0 (Jet 3.0) or 2.0 (Jet 2.x). However, you can open and use version 2.0 .MDB and Access 95 .mdb files with Access 97. Many Access 2.0 users were forced to forego the transition to Access 95 because of limited deployment of Windows 95 by large organizations. Consequently, this chapter begins by addressing the ramifications of continuing to use version 2.0 .MDB files with Access 97 and then explains how to convert your existing .MDB files to Access 95's 32-bit .mdb structure. The chapter concludes with a discussion of the relatively minor changes involved in converting Access 95 files to Access 97 format.

Using Access 2.0 Application .MDB Files with Access 97

Access 97 is designed to be backward compatible with Access 2.0 .MDB files. Nevertheless, the compatibility is not total. The following list describes the principal limitations of running version 2.0 .MDB files with Access 95:

- You cannot save any changes you make to the design of any object contained in a version 2.0 .MDB file.

Oracle Ex. 1009, p. 1159
Oracle v. DeBellis
IPR2024-00221

- You cannot change ownership of, or permissions for, objects contained in a version 2.0 .MDB file.
- `DoMenuItem` actions that refer to menu choices that have changed in Access 97 produce an unexpected result. Access replaces `DoMenuItem` action with the `RunCommand` action in macros and Access VBA code.
- `SendKeys` operations that execute Access 97 menu choices and make selections in dialogs are likely to fail due to changes in the Access 97 menu structure and the design of Access 97 dialogs.
- Variable names that conflict with the names of new VBA 5.0 reserved words result in failure of Access 97 to compile modules and to execute procedures containing the conflicting variables.
- Access VBA statements that use the dot (.) operator to refer to a field of a `Recordset` object or a member of a collection fail. Use the bang (!) operator, as in `Orders!OrderID` or `Forms!Orders`.

If you don't need to change the design of your Access 2.0 application and the application does not contain any of the specific problem areas in the preceding list, you probably can continue to use the Access 2.0 application with Access 97 and share the application with other users of Access 97 and earlier versions of Access. The message shown in Figure 33.1 appears when you first open an Access 2.0 application in Access 97.

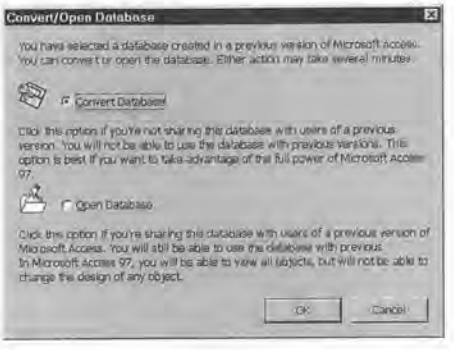


FIG. 33.1 The dialog that appears when you first open an Access 2.0 .MDB or an Access 95 .mdb file in Access 97.

To attempt to run an Access 2.0 application under Access 97, choose the Open option in the dialog of Figure 33.1. The status bar displays `Opening`, then `Compiling`, and finally the message shown in Figure 33.2 appears advising that you can't change the design of objects in the database. The opening and compiling process, which enables Access 2.0 and 95 applications, makes Access Basic modules of Access 2.0 or Access VBA of Access 95 compatible with VBA 5.0. Fortunately, the opening and compiling process only occurs once, but the message box of Figure 33.2 continues to appear each time you open the database.

Oracle Ex. 1009, p. 1160
Oracle v. DeBellis
IPR2024-00221

Note

If you make changes to an Access 2.0 or 95 application, the enabling process reoccurs when you first open the application. Make sure you compile all of the modules in your earlier version .MDB file before opening it in Access 97.



FIG. 33.2 The recurring message that appears when you open an enabled Access 2.0 or 95 .MDB file in Access 95.

If your application contains code that Access VBA can't handle, you're likely to receive the message shown in Figure 33.3. To find the offending code, open a module and click the Compile Loaded Modules button. As an example, Access 2.0's Nwind.mdb sample database contains a subprocedure, `ShowEvents`, in the Utility Functions module that the Access 97 VBA interpreter won't compile. Each line that contains the `Events` variable name is colored red and the message box shown in Figure 33.4 appears. You can't save corrections to Access Basic code in Access 97, so you must make any required changes to the unconverted .MDB file in the appropriate version of Access. In the case of the `ShowEvents` subprocedure, you change the variable name from `Event` to `strEvent` in three locations. You then recompile the code changes in the original version of Access and try opening the .MDB file again in Access 97.



FIG. 33.3 The message that appears when the VBA 5.0 interpreter can't handle your Access Basic code.

Oracle Ex. 1009, p. 1161
Oracle v. DeBellis
IPR2024-00221



FIG. 33.4 The message box that attempts to identify the problem with Access Basic code.

Using Access 1.x and 2.0 Data .MDB Files with Access 97

If you use the recommended split-database method (storing application objects and data objects in two separate .MDB files), you can convert your Access 1.x or 2.0 application .MDB file to Access 97, but leave the data .MDB as a version 1.x or 2.0 .MDB file. Users of 16-bit Access, Access 95, and Visual Basic 3.0, 4.0, or 5.0 can share the linked data .MDB files. When you attach tables contained in Access 1.x, 2.0, or 95 .MDB files, you don't receive a warning message. You cannot use Access 95 features, such as Briefcase replication, with .MDB files created with Access 2.0 or earlier.

> **Note**
>
> It is a relatively easy process to split Access 1.x or 2.0 applications into separate data and application .MDB files. (Access 95 and 97 include the Database Splitter Add-In that automates this process; prior versions of Access do not offer this feature.) In your prior version of Access, create a new .MDB file, then import the tables from your existing .MDB file. Verify that the imported tables contain valid data, then delete the tables from your original .MDB file. Move the table to its final location, such as on a network server. Then attach the tables to the application .MDB file.



◀◀ See "Splitting Databases for File Sharing," p. 873

If any of the conditions in the following list apply to your Access applications, maintain your files in their earlier Access format:

- Some of the users of your Access applications have not upgraded to Windows 95 or Windows NT, so those users are unable to run the Access 97 retail or run-time versions.

Oracle Ex. 1009, p. 1162
Oracle v. DeBellis
IPR2024-00221

- You are converting a series of different application .MDB files from version 1.x or 2.0 to Access 97 and need to maintain data compatibility with all application .MDBs until the conversion process is complete.
- You are sharing the database .MDB with a Visual Basic 3.0 application.

If you've developed several workgroup applications using earlier versions of Access, you must maintain the shared data .MDB file in its original format until you have converted all of the application databases to Access 97. There is little or no performance penalty for attaching tables and maintaining the original data .MDB file format.

Caution

Do not convert existing shared Access 1.x, 2.0, or 95 SYSTEM.MDA or System.mdw files to Access 97 format. If you do, users of prior versions of Access will not even be able to launch Access. (They receive an `Unable to open 'filename.mda'. It may not be an Access database or the file may be corrupt.` message) Access system files are attached, so you can use existing workgroup system files with Access 97.

Converting Access 1.x and 2.0 Files to Access 95

If you elect to convert your earlier versions of Access application .MDB files to Access 95 and all of your users have not converted to Access 95, you need to maintain two versions of the application .MDB file. If the application is mature and does not require significant maintenance, temporarily supplying a 32-bit Access 97 version to users of Windows 95 and Windows NT 4.0, and a 16-bit Access 2.0 or earlier version to users of Windows 3.1+ is not likely to present a problem.

If you're in the development or early roll-out phase, however, consider completion of the application in Access 2.0. Make sure to observe the syntax rules for VBA 5.0 while writing your Access Basic code; doing so minimizes the trauma when converting your Access 2.0 application to Access 97.

Converting versus Importing .MDB Files

You have the following options for converting version 1.x or 2.0 application, data, or combined application and data .MDB files to Access 97 format:

Tip

If you encounter problems with permissions to modify objects in your converted .mdb file, try the import method so that you become the owner of the objects in the database.

- Open the file in Access 97 and accept the default Convert Database option. Alternatively, close any open database, then choose Tools, Database Utilities, Convert. This is the fastest method of performing the conversion. The ownership of objects

Oracle Ex. 1009, p. 1163
Oracle v. DeBellis
IPR2024-00221

in the converted database does not change. Access 97 requires you to specify a different file name or directory for the converted .mdb file so you don't overwrite the existing version.

- Choose File, Get External Data, Import to import all of the objects from the version 1.x or 2.0 .MDB to a newly created Access 97 .mdb file. When you import the objects, you become the owner (Creator) of the objects.

Tip

When importing an entire database, you can select all of the objects for import at once by clicking each tab (Tables, Queries, Forms, and so on) and clicking the Select All button for each class of object. Once you've selected all of the objects to be imported, click the OK button.

The database object permissions assigned to users and groups are not affected by either converting or importing the objects. Only ownership of the objects is affected.

Handling Conversion and Import Errors

Small Access 1.x and 2.0 applications, especially applications without Access Basic code, are likely to convert to Access 97 without problems. However, large applications that contain substantial amounts of code or use code contained in Access libraries or custom .DLLs, generate errors during the conversion process. The following sections describe the most common conversion errors and how to fix them.

Converting from Win16 to Win32 Function Calls. If your application makes use of calls to the Windows API, you must convert the function declarations from the 16-bit (Win16) to the 32-bit (Win32) version. You cannot call 16-bit functions from 32-bit code and vice versa. You receive a `Calls to 16-bit DLLs won't work under Windows 95 or Windows NT` message when the VBA interpreter encounters a 16-bit Windows API declaration.

The Win32 API functions with arguments of the String data type come in two types: ANSI (suffix `A`) and Unicode (suffix `W`, for wide); the ANSI versions are used by Access 97. Most Windows API function arguments that were of the **`Integer`** data type in the Win16 version must be declared as **`Long`** in the Win32 version. Examples include all Win32 handles, such as `hWnd` and `hDC`, and values of **`Integer`** fields in structures. To avoid the need to change all function calls to Win32 API functions in your Access VBA code, you alias the function calls, as shown in the following example:

```
Declare Function OriginalName Lib "lib32" _
Alias "OriginalName[A]" ([ArgumentList]) As Datatype
```

`OriginalName` is the name of the Win16 function in your Access VBA function calls. Win32 libraries, for the most part, have a `32` suffix, as in Kernel32.exe and GDI32.exe. In the majority of cases, you need only include the `A` suffix in the **`Alias`** name if **`As String`** appears in *`ArgumentList`*. For most Win32 functions, ***`Datatype`*** is **`Long`**.

Oracle Ex. 1009, p. 1164
Oracle v. DeBellis
IPR2024-00221

The Win API Viewer, which is included with the Office 97 Developer Edition (ODE) and Visual Basic 4+, lets you copy `Declare` statements for Win32 function calls to the Clipboard, then paste them into the Declarations section of your modules. If you don't have the ODE or Visual Basic 4+, you must either have the Win32 Software Development Kit (SDK) or guess the correct Win32 API function name and data types, then try compiling and executing your code.

◀◀ See "Using the 32-Bit Windows API Viewer," p. 1147



Converting and Adding References to Libraries and Add-Ins. In addition to converting your application .MDB file to Access 97, you also must convert any Access 2.0 custom libraries and add-ins used by your application to 32-bit versions. Converting your own libraries and add-ins follows the same process as converting .MDB files. If you use third-party libraries, you'll likely need updated Access 97 versions; the source code of many third-party libraries is password-protected.

> **Note**
>
> If you are using only a few functions in a library, such as the WIZLIB.MDA library included with Access 2.0, consider creating a new library or incorporating the function code in your Access 97 application. Doing so eliminates dealing with problems converting library functions you don't need. Copy the required function(s) to the Clipboard and paste the code into an Access 97 module. (You can run prior versions of Access and Access 97 simultaneously.)

Access 2.0 and earlier use the `[Libraries]` section of MSACC??0.INI to attach libraries to any application opened in Access. Access 95 and 97 use references to 32-bit libraries in place of entries in MSACC??0.INI. To add a reference to a library, such as Access 97's Utility.mda, follow these steps:

1. Open a module and choose Tools, References to open the References dialog.
2. Click the Browse button to open the Add Reference dialog.
3. Select Add-ins (*.mda) in the Files of Type drop-down list.
4. Maneuver to the folder that contains your library, select the library file, and click the OK button to add the reference at the bottom of the Available References list.
5. When you close and reopen the References dialog, the new reference appears as the last active reference. Figure 33.5 shows Access 97's Utility.mda library added to the Available References list.

Oracle Ex. 1009, p. 1165
Oracle v. DeBellis
IPR2024-00221

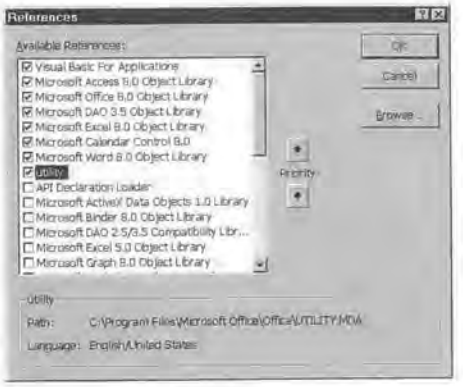


FIG. 33.5 References to Access 97 library databases appear below Access 97's default references and any references you added previously.

Access 97 library references are stored in your .mdb file; thus, you must create a reference to a library in each .mdb file that uses the library. Unlike `[Libraries]` entries in MSACC??0.INI, the full path to the library is hard-coded. You may need to alter the reference for users who don't install your application in the folder specified by the reference in your .mdb file. Access 97 lets you modify references with Access VBA; all members of Office 97 expose a `References` collection.

Hard-coded reference paths are a compelling reason to move existing library code to your Access application .mdb if you only use a few library functions. If your library code does not contain visible objects, other than message boxes and modal dialogs, an alternative is to convert the library to an in-process Visual Basic 32-bit ActiveX DLL that you manage with Automation code.

> **Note**
>
> Access 97, unlike Access 2.0 and earlier, allows duplicate function names in libraries. If you have duplicate function names in libraries, your application calls the function in the library with the highest priority in the Available References list of the References dialog.

Access 2.0 used the `[Menu Add-Ins]` section of MSACC20.INI to specify the function name of the entry point for an add-in and the `[Libraries]` section to attach the .MDA file that contains the add-in. These entries are replaced by the Registry entries in Access 95 and 97. If your Access 97 application uses add-ins, a record in the USysRegInfo table of the add-in is required to make the add-in visible to the Add-In Manager.

You can import the USysRegInfo table of one of the Access 97 wizards into your add-in, then modify the entries as required for your add-in. When you use the Add-In Manager to install the add-in, an entry is appended to the MenuAdd-Ins section of the Registry entries for Access 97. Figure 33.6 shows the data for Access 97's Add-In Manager add-in. Unlike references to library databases, add-ins are available to all databases you open in Access 97.

Oracle Ex. 1009, p. 1166
Oracle v. DeBellis
IPR2024-00221

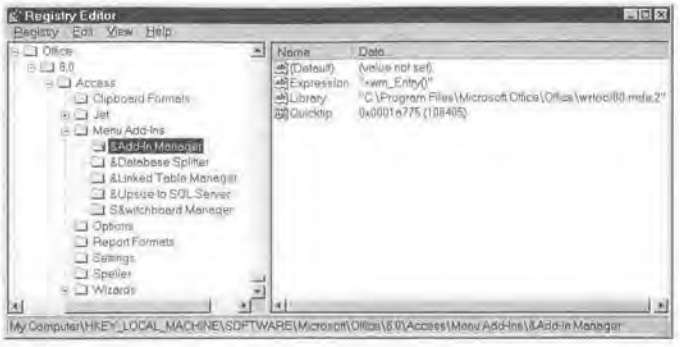


FIG. 33.6 The Add-In Manager appends entries to the MenuAdd-Ins section of the Registry.

The 32-Index Limit on Tables. Access 97 imposes a limit of 32 indexes per table. Each relationship between tables creates an index on the two tables that participate in the relationship, a feature introduced in Access 95. If you have a very complex database with many relationships between tables, you may exceed the 32-index per table limit.

Although your Access 2.0 table may have less than 32 indexes, when you convert the table to Access 97, the additional indexes for relationships may exceed the limit of 32. In this case, you either cannot convert the database or, if the database converts, you lose indexes. Your only option in this situation is to reduce the number of indexes on your table in your prior version of Access, then try the conversion process again.

Converting 16-bit OLE Controls to 32-bit ActiveX Controls. If your Access 2.0 application uses 16-bit OLE Controls and you have the 32-bit ActiveX version of the original control, Access 97 automatically updates the control to the 32-bit ActiveX version. The ActiveX control must be registered for the conversion to work. If you don't have a 32-bit ActiveX version of the control, an empty control container appears on your form.

> **Note**
>
> Access 97 does not include a 32-bit ActiveX version of the Data Outline Control provided with Access 95. The 32-bit Data Outline OLE Control of Access 95 works, but is not supported by Microsoft, in Access 97. If the Access 2.0 application you want to convert to Access 97 uses the 16-bit Data Outline (Navigator) Control, you must acquire the Access 95 Developer's Toolkit (ADT) to obtain the 32-bit version and a license to distribute the Data Outline Control.

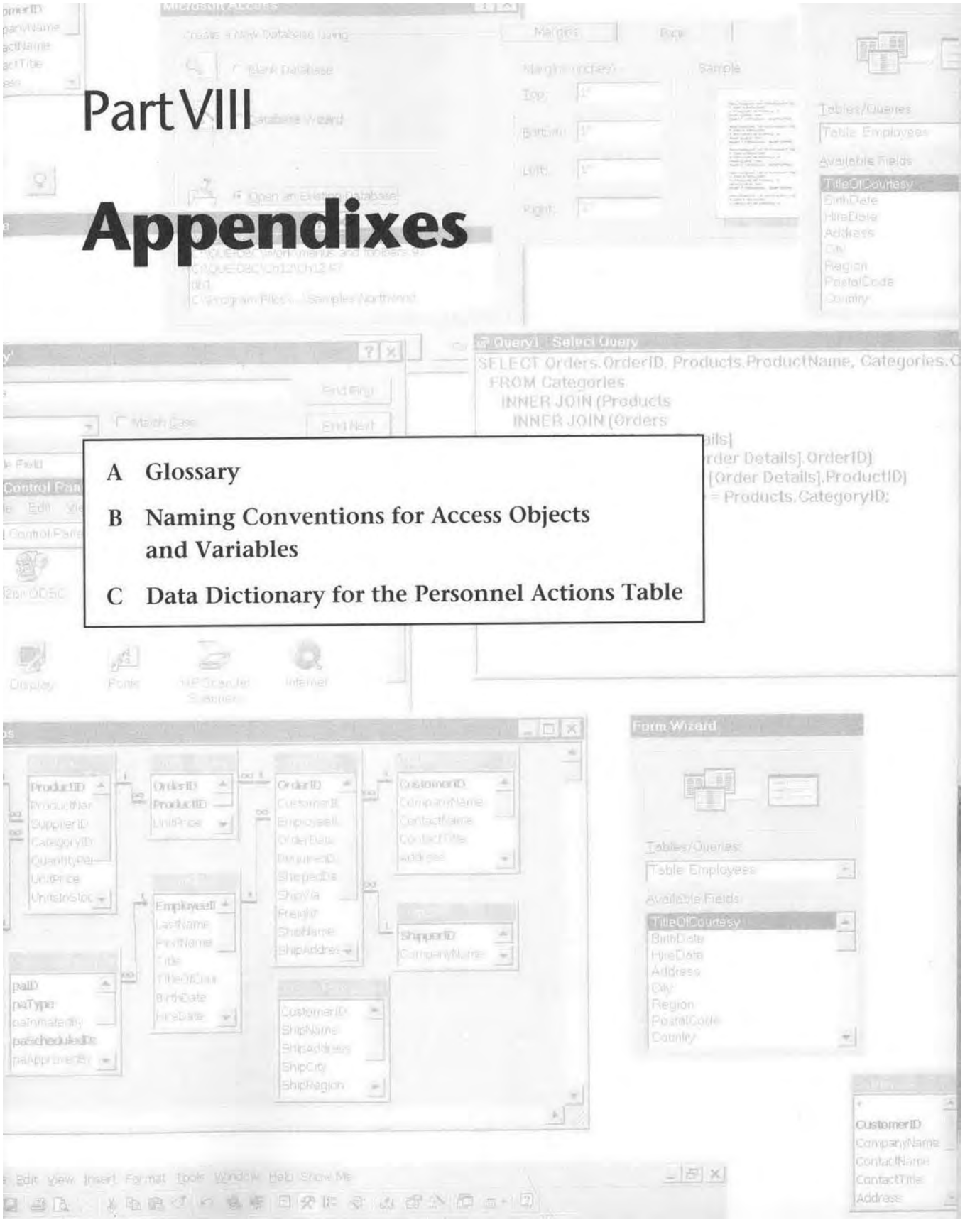
Converting Access 95 Databases to Access 97 Format

The binary structures of Access 95 and 97 files are quite similar, so conversion issues between Access 95 and 97 applications are fewer than those between Access 2.0 and 95. Following are the most common issues you might encounter when upgrading your Access 95 .mdbs to Access 97:

Oracle Ex. 1009, p. 1167
Oracle v. DeBellis
IPR2024-00221

- Access 97 converts all `DoCmd.DoMenuItem` statements to `DoCmd.RunCommand` statements. Access 97 also performs this conversion for DoMenuItem actions in macros. Problems may occur in the conversion of custom toolbars and, to a lesser degree, custom menus created in Access 95 to `CommandBar` objects. If conversion problems occur, you must rebuild or repair your custom toolbars and menus in Access 97.
- Access 97 can't open a database replica created in Access 95. You must convert the Design Master and replicas to Access 97 format.
- To improve performance, many optional parameters of Access VBA methods are strongly-typed in Access 97, meaning that use of the **`Variant`** data type is not allowed. You receive a `Data Type Mismatch` error when the VBA 5.0 interpreter encounters such parameters. Use the Object Browser to determine the required data type.
- A few error codes have changed. If you use numeric error code values in error-handling routines, you might need to change the numeric values.
- Access 95's DAO 3.0 caused SQL statements that modify data to execute as implicit transactions. Using the `dbFailOnError` constant caused failed SQL operations to be rolled back. To improve performance, DAO 3.5 no longer provides implicit transactions. Use the `BeginTrans`, `CommitTrans`, and `Rollback` methods to create explicit transactions in your VBA 5.0 code.

Oracle Ex. 1009, p. 1168
Oracle v. DeBellis
IPR2024-00221



Part VIII

Appendixes

A Glossary

B Naming Conventions for Access Objects and Variables

C Data Dictionary for the Personnel Actions Table

Oracle Ex. 1009, p. 1169
Oracle v. DeBellis
IPR2024-00221

Appendix A
Glossary



accelerator key A key combination that provides access to a menu choice, macro, or other function of the application in lieu of selection with the mouse, usually by combining Alt+*key*. An accelerator key is identified on menus by an underlined character. It is sometimes (incorrectly) called a *shortcut key*, but shortcut keys usually consist of Ctrl+*key* combinations.

Access Control List (ACL) Part of Windows NT's security description that controls access to a Windows NT object, such as a file. The owner of an object can change access control entries in the list to grant or revoke permissions (access rights) for the object.

Access Developer's Toolkit See *ADT* and *ODK*.

Access SQL The dialect of ANSI SQL used to write queries in all versions of Microsoft Access. For the most part, Access SQL complies with ANSI SQL-92. Access SQL offers additional features, such as including user-defined functions within queries.

access token A Windows NT object that identifies a logged on (authenticated) user. The access token contains the user's security ID (SID), the groups to which the user belongs, and other security information. See *SID*.

activation An OLE 2.0 term meaning to place an object in a running state, which includes binding the object, or to invoke a method of the object. See also *binding*.

active In Windows, the currently running application or the window to which user input is directed; the window with the focus. See also *focus*.

ActiveMovie A Microsoft video technology that provides synchronized images and sound under Windows 95 and Windows NT 4.0, intended as a replacement for 32-bit Video for Windows.

ActiveX A Microsoft trademark for a collection of technologies based on the Common Object Model (COM) and Distributed Common Object Model (DCOM). See *COM* and *DCOM*.

Oracle Ex. 1009, p. 1170
Oracle v. DeBellis
IPR2024-00221

ActiveX components A replacement term for OLE Automation mini-servers and in-process servers, also called *Automation servers*. See *OLE Automation*.

ActiveX Controls Insertable objects supplied in the form of OCX files that, in addition to offering a collection of properties and methods, also fire events. ActiveX Controls are lightweight versions of OLE controls that also use the .OCX file extension. See *OLE controls*.

ActiveX data objects High-level data objects, similar in concept to Access 97's Data Access Object (DAO) and Remote Data Object (RDO), that use Microsoft's new OLE DB (OLE Database) technology to access data from a variety of data sources, including text files and mainframe databases. See *OLE DB*.

ActiveX documents Files that can be inserted into the Microsoft Binder, such as files created by Microsoft Excel 7+ and Word 7+, as well as displayed in their native format in Internet Explorer 3+. ActiveX documents originally were called *Document Objects* or *DocObjects*.

ActiveX scripting Another name for Visual Basic, Scripting Edition (VBScript), a simplified version of VBA designed for client-side automation of Web pages.

ActiveX server framework ActiveX scripting for creating server-side Internet and intranet applications. Unlike ActiveX scripting, the ActiveX Server Framework (code-named Denali) allows file and other low-level operations.

add-in A wizard (such as the Query Wizard) or builder (such as the Menu Builder) that helps users of Access create or run database applications. You use Access 97's Add-In Manager to install wizards and builders (choose Tools, Add-Ins). See also *builder*.

address The numerical value, usually in hexadecimal format, of a particular location in your computer's random-access memory (RAM).

ADT Abbreviation for the Access Developer's Toolkit for Access 2.0 and Access 95 that allowed distribution of files needed to run (but not design) Access 2.0 and Access 95 applications. For Access 97, the ADT is replaced by the Office Developer's Kit (ODK), which provides the same features as the Access 95 ADT, plus developer products for other Office applications. See *ODK*.

aggregate functions The ANSI SQL functions `AVG()`, `SUM()`, `MIN()`, `MAX()`, and `COUNT()` and Access SQL functions `StDev()`, `Var()`, `First()`, and `Last()`. Aggregate functions calculate summary values from a group of values in a specified column. They are usually associated with `GROUP BY` and `HAVING` clauses. See also *domain aggregate functions*.

aggregate object An OLE 2.0 term that refers to an object class that contains one or more member objects of another class.

alias A temporary name assigned by Access to a table in a self join, to a column of a query, or to rename a table, implemented by the `AS` reserved word in ANSI SQL. You can use `AS` to rename any field or table with Jet SQL. **`Alias`** is also an embedded keyword

Oracle Ex. 1009, p. 1171
Oracle v. DeBellis
IPR2024-00221

option for the VBA **Declare** statement. The **Alias** keyword is used to register prototypes of DLL functions so that the function can be called from programs by another name. Aliasing the ANSI versions of 32-bit Windows API functions to function names without the A suffix is common when converting Access 1.x and 2.0 applications to Access 97, which uses Unicode strings.

ANSI An abbreviation for the American National Standards Institute. ANSI in the Windows context refers to the ANSI character set that Microsoft decided to use for Windows (rather than the IBM PC character set that includes special characters such as those used for line drawing, called the *OEM character set*). The most common character set is *ASCII* (the *American Standard Code for Information Interchange*), which for English alphabetic and numeric characters is the same as ANSI. Windows 95 and Windows NT include both ANSI (suffix A) and Unicode (suffix W) versions of Windows API functions. See *ASCII* and *Unicode*.

API An abbreviation for Application Programming Interface. Generically, a method by which a program can obtain access to or modify the operating system. In 32-bit Windows, the 1,000 or so functions provided by Windows 95 and Windows NT DLLs that allow applications to open and close windows, read the keyboard, interpret mouse movements, and so on. Programmers call them *hooks*. VBA provides access to these functions with the Declare statement. See also *DLL*.

applet A Windows application that is supplied as a component of another Windows application, rather than a retail product. The Notepad, WordPad, and Character Map applications supplied with Windows 95 are examples of applets.

application The software product that results from the creation of a program, often used as a synonym for the programming (source) code that creates it. Microsoft Word, Microsoft Excel, WordPerfect for Windows, and Lotus 1-2-3 are called *mainstream Windows productivity applications* in this book. Applications are distinguished by the environment for which they are designed (such as Windows, DOS, Macintosh, and UNIX) and their purpose. Windows applications carry the DOS executable file extension exe.

Application Close button The small, square button with an "X" caption at the extreme right of the title bar of an application running in Windows 95. Clicking the Application Close button closes the running application.

Application Control menu box The small square button with a miniature application icon at the extreme left of the title bar of an application. Clicking the Application Control menu box displays the Application Control menu. Double-clicking the Application Control menu box closes the application.

argument Data supplied to a function and upon which the function acts or uses to perform its task. Arguments are enclosed in parentheses. Additional arguments, if any, are separated by commas. Arguments passed to procedures usually are called *parameters*.

Oracle Ex. 1009, p. 1172
Oracle v. DeBellis
IPR2024-00221

array An ordered sequence of values (elements) stored within a single named variable, accessed by referring to the variable name with the number of the element (index or subscript) in parentheses, as in `strValue = strArray(3)`. Arrays in VBA may have more than one dimension, in which case access to the value includes indexes for each dimension, as in `strValue = strArray(3,3)`.

ASCII Abbreviation for the American Standard Code for Information Interchange. A set of standard numerical values for printable, control, and special characters used by PCs and most other computers. Other commonly used codes for character sets are ANSI (used by Windows 3.1+), Unicode (used by Windows 95 and Windows NT), and EBCDIC (Extended Binary-Coded Decimal Interchange Code, used by IBM for mainframe computers). See *Unicode*.

assign To give a value to a named variable.

asynchronous A process that can occur at any time, regardless of the status of the operating system or applications that are running.

attached table A table that is not stored in the currently open Access database (native or base table), but which you can manipulate as if the table were a native table. In Access 95 terminology, an attached table is a linked table. See *linked table*.

authentication The process of verifying a user's login ID and password.

automation An ActiveX and OLE 2.0 term that refers to a means of manipulating another application's objects. See also *OLE Automation*.

automation client An ActiveX- or OLE 2-compliant Windows application with an application programming (macro) language, such as VBA, that is capable of referencing and manipulating objects exposed by (OLE) Automation servers.

automation server Technically, any COM- or OLE 2-compliant Windows application that supports Automation operations by exposing a set of objects for manipulation by Automation client applications. This book restricts the term automation server to applications that are not OLE 2+ full servers, but expose application objects. Access 97 is an example of an automation server.

AutoNumber An Access 95 replacement for the Counter field data type of Access 1.x and 2.0. AutoNumber fields may be of the Increment or Random type. Fields of the Increment AutoNumber field data type usually are used to create primary keys in cases where a unique primary key cannot be created from data in the table.

AutoPlay A feature of Windows 95's and Windows NT 4.0's CD-ROM file system (CDFS) that automatically executes a program on the CD-ROM when inserted into the CD-ROM drive.

back up To create a file (backup file) that duplicates data stored in one or more files on a client or server computer.

Oracle Ex. 1009, p. 1173
Oracle v. DeBellis
IPR2024-00221

background In multitasking computer operations, the application or procedure that is not visible on-screen and that does not receive user generated input. In Windows, an application that is minimized and does not have the focus is in the background.

base date A date used as a reference from which other date values are calculated. In the case of VBA and SQL Server, the base date is January 1, 1900.

base tables The permanent tables from which a query is created. A synonym for underlying tables. Each base table in a database is identified by a name unique to the database. Access also uses the term *base table* to refer to a table in the current database in contrast to a linked (attached) table. See *linked table*.

batch A group of statements processed as an entity. Execution of DOS batch files, such as AUTOEXEC.BAT, and SQL statements are examples of a batch process.

BDC An abbreviation for Backup Domain Controller, a Windows NT server that provides an alternative source of authentication for network users. Account and group information from a Primary Domain Controller (PDC) is replicated periodically to each BDC in the domain. See *PDC*.

binary file A file whose content does not consist of lines of text. Executable (.exe), dynamic link library (.dll), and most database files are stored in binary format.

binary string A string consisting of binary, not text, data that contains bytes outside the range of ANSI or ASCII values for printable characters. Access 97 requires that you store binary strings as arrays of the Byte data type to avoid problems with Unicode/ANSI conversion.

binding In Access, attaching a Form or Report object to a table, or a control object to a field of a table or the column of a query result set. The bound Form or Report object determines the current record of the table or the bound control object reflects the value of the data cell or field of the current record or row.

bit The smallest piece of information processed by a computer. A bit, derived from the contraction of BInary digiT (or Binary digIT) has two states: on (1) or off (0). Eight bits make up a *byte*, and 16 bits combined is called a *word*.

bitmap The representation of a screen or printed image, usually graphic, as a series of bytes.

bitwise A process that evaluates each bit of a combination, such as a byte or word, rather than processing the combination as a single element. Logical operations and masks use bitwise procedures.

blitting The process of using the `BitBlt()` function of Windows' GdI32.exe to modify a bitmap using bit block transfer.

Boolean A type of arithmetic in which all digits are bits; that is, the numbers may have only two states: on (true or 1) or off (false or 0). Widely used in set theory and computer programming, Boolean, named after the mathematician George Boole, also is used

Oracle Ex. 1009, p. 1174
Oracle v. DeBellis
IPR2024-00221

to describe a VBA data type that may only have two states: true or false. In VBA, True is represented by `&HFF` (all bits of an 8-bit byte set to 1) and False by `&H0` (all bits set to 0).

bound See *Binding and Object frame.*

break To cause an interruption in program operation. Ctrl+C is the standard DOS break key combination, but it seldom halts operation of a Windows application. Esc is more commonly used in Windows to cause an operation to terminate prior to completion.

breakpoint A designated statement that causes program execution to halt after executing the statement preceding it. Breakpoints may be toggled on or off by choosing Run, Toggle Breakpoint in Access, or by pressing the F9 function key.

BRI An abbreviation for Basic Rate Interface, the standard ISDN service for business and residential Internet connections. BRI has two 56Kbps B (bearer) channels and one 16Kbps D (data) channel, providing a maximum bandwidth of 112Kbps. See *ISDN* and *PRI.*

Briefcase replication A feature of Access 97 running under Windows 95 that permits the creation of Access replication sets stored in Windows 95 Briefcase folders, which can be updated by mobile users. Subsequently, the briefcase replicates are used to update the design-master replica to synchronize the design-master replica with the contents of the briefcase replicas. See *design-master replica.*

buffer An area in memory of a designated size (number of bytes or characters) reserved, typically, to hold a portion of a file or the value of a variable. When string variables are passed as arguments of DLL functions, you must create a buffer of sufficient size to hold the returned string. This is accomplished by creating a fixed-length string variable of the necessary size, using the `String()` function, prior to calling the DLL function.

builder A component of Access that provides assistance in creating expressions (Expression Builder) or controlling objects.

built-in functions Functions that are included in a computer language and need not be created by the programmer as user-defined functions.

business rules A set of rules for entering data in a database that are specific to an enterprise's method of conducting its operations. Business rules are in addition to rules for maintaining the domain and referential integrity of tables in a database. Business rules most commonly are implemented in a three-tier client-server database environment. See *three-tier.*

cache A block of memory reserved for temporary storage. Caches usually store data from disk files in memory to make access to the data faster. By default, Windows 95 caches all disk read and write operations.

caption The title that appears in the title bar of a window. Access calls the text of a label, check box, frame, and command or option button control object the `Caption` *property.*

Oracle Ex. 1009, p. 1175
Oracle v. DeBellis
IPR2024-00221

caret The term used by Windows to indicate the cursor used when editing a text field, usually shaped as an I-beam. The caret, also called the *insertion point*, can be positioned independently of the mouse pointer.

Cartesian product Named for René Descartes, a French mathematician. Used in `JOIN` operations to describe all possible combinations of rows and columns from each table in a database. The number of rows in a Cartesian product is equal to the number of rows in table 1 times that in table 2 times that in table 3, and so on. Cartesian rows that do not satisfy the `JOIN` condition are disregarded.

cascading deletion A trigger that deletes data from one table based on a deletion from another table to maintain referential integrity. Usually used to delete detail data (such as invoice items) when the master record (invoice) is deleted. Access 2+ provides cascading deletion as an optional component of its referential integrity features. See *referential integrity*.

case sensitivity A term used to define whether the interpreter or compiler treats lowercase and uppercase letters as the same character. Most are case-insensitive. C is an exception; it is case-sensitive, and all of its keywords are lowercase. Many interpreters—VBA included—reformat keywords to its standard: a combination of uppercase and lowercase letters. VBA does not distinguish between uppercase and lowercase letters used as names for variables.

CDFS The 32-bit CD-ROM file system shared by Windows NT and Windows 95.

channel In Windows, ordinarily refers to a unique task ID assigned to a dynamic data exchange (DDE) conversation. Channel IDs are Long integers under Windows 95 and Windows NT. Also used to identify an I/O port in mini- and mainframe computers.

check box A windows dialog and Access control object that consists of a square box and an associated caption. A diagonal cross or other mark in the box is created or erased (toggled) by alternate clicks on the box or the label with the mouse or by pressing an assigned hotkey.

child In Windows, usually an abbreviation for an MDI child window. Also used in computer programming in general to describe an object that is related to but lower in hierarchical level than a parent object.

chunk A part of either a RIFF or standard MIDI file that is assigned to a particular function and may be treated as a single element by an application. VBA uses the term *chunk* to refer to a part of any file that you read or write with the `GetChunk` and `AppendChunk` methods. See *RIFF*.

class identifier See *CLSID*.

clause The portion of an SQL statement beginning with a keyword that names a basic operation to be performed.

client The device or application that receives data from or manipulates a server device or application. The data may be in the form of a file received from a network file server,

Oracle Ex. 1009, p. 1176
Oracle v. DeBellis
IPR2024-00221

an object from an ActiveX component or OLE server, or values from a DDE server assigned to client variables. See *automation client*.

Clipboard Windows' temporary storage location for text and graphic objects, as well as Access objects, such as control objects, forms, tables, reports, and so on. The Clipboard is the intermediary in all copy, cut, and paste operations. You can view and save the contents of the Clipboard using the Program Manager's Clipboard applet.

CLSID An identification tag that is associated with an OLE 2.0 object created by a specific server. CLSID values appear in the Registry and must be unique for each ActiveX component or OLE 2.0 server and each type of object that the server can create. See *Registry*.

clustered index An index in which the physical record order and index order of a table are the same.

clustering A server architecture that emulates multiprocessing by interconnecting two or more individual computers in order to share the application processing load. Microsoft's future clustering technology for Windows NT now carries the codename *Wolfpack*. A number of third parties offer proprietary clustering hardware and software for Windows NT 4.0 Server.

code Short for *source code*. The text you enter in your program to create an application. Code consists of instructions and their parameters, functions and their arguments, objects and their events, properties and methods, constants, variable declarations and assignments, and expressions and comments.

code template Self-contained groups of modules and resources that perform a group of standard functions and that may be incorporated within other applications requiring these functions, usually with little or no modification.

Code window In Access, the window that appears when you select Module from the Database window or click the Ellipsis button of an event property to create or edit an event-handling subprocedure. Also called the *code editing window*.

coercion The process of forcing a change from one data type to another, such as Integer to Text.

collection A group of objects of the same class that are contained within another object. Collections are named as the plural of their object class. As an example, the Forms and Reports collections are groups of Form and Report objects contained in the Database object.

color palette A means of establishing a foreground or background color in Windows by selecting a color from those displayed with the mouse. The color palette then converts the selection to the standard Windows RGB (red/green/blue) color format. The color palette provides the set of colors for graphic objects of 256 colors or less. Access 2+ allows you to specify a particular palette for individual forms. Also called *palette* or *Windows palette*.

Oracle Ex. 1009, p. 1177
Oracle v. DeBellis
IPR2024-00221

COM An acronym for Component Object Model, the name of Microsoft's design strategy to implement ActiveX and OLE 2+. Distributed COM (DCOM) allows networked and cross-platform implementation of ActiveX and OLE 2+ operations, and (OLE) Automation. See *DCOM*.

combo list A Windows object that combines text box and list elements into a single object. In Access, combo lists are of the drop-down type by default. The list element of a drop-down combo list appears when a downward-pointing arrow to the right of the text box is clicked.

command A synonym for instruction. Specifies an action to be taken by the computer.

command button A Windows object that causes an event when clicked. Command buttons are ordinarily a gray rectangle containing a caption and surrounded by a border.

comment Explanatory material within source code not designed to be interpreted or compiled into the final application. In VBA, comments are usually preceded by an apostrophe (`'`), but can also be created by preceding them with the `Rem` keyword.

common dialog A standardized dialog box, provided by Windows 95 and Windows NT, that may be created by a Windows API function call to functions contained in Cmdlg32.dll. Common dialogs include FileOpen, FileSave, Print and Printer Setup, ColorPalette, Font, and Search and Replace. Using the common dialogs in Access applications requires that you use the `Declare` statement to create function prototypes for the functions in Comdlg32.dll that you plan to use. The Comdlg32.ocx control, included with the Office Developer's Kit, lets you implement most of the common dialogs without the necessity of calling Comdlg32.dll functions.

Common User Access See *CUA*.

comparison operators See *operator*.

compile To create an executable or object (machine-language) file from source (readable) code. In Access, *compile* means to create pseudo-code (tokenized code) from the VBA source code you write in the code editing windows.

Component Object Model See *COM*.

composite key or index A key or index based on the values in two or more columns. See also *key* and *index*.

composite menu A menu that includes menu choices from an OLE 2.0 server application that uses in-place (in-situ) activation (editing). Creating a composite menu also is called *grafting a menu*.

composite moniker The location within a container document or object where the compound document is located.

Oracle Ex. 1009, p. 1178
Oracle v. DeBellis
IPR2024-00221

compound In computer programming, a set of instructions or statements that requires more than one keyword or group of related keywords to complete. `Select Case...Case...End Select` is an example of a compound statement in VBA.

compound document A document that contains OLE objects created by an application other than the application that originally created or is managing the document.

concatenation Combining two expressions, usually strings, to form a longer expression. The concatenation operator is `&` in SQL and VBA, although VBA also permits the + symbol to be used to concatenate strings.

concurrency The condition when more than one user has access to a specific set of records or files at the same time. Concurrency is also used to describe the ability of a database management system to handle simultaneous queries against a single set of tables.

container An object or application that can create or manipulate compound documents.

context switching The process of saving an executing thread or process and transferring control to another thread or process. Windows NT 4.0's context switching—one of the major bottlenecks in COM operations—is substantially faster than in Window NT 3.x.

control A synonym for a dialog object in Access. Controls include labels, text boxes, lists, combo lists, option buttons, and command buttons. Access 97 also provides compatibility with ActiveX Controls (formerly OLE Controls).

control array In Visual Basic, the term given to multiple controls on a single form with the same `Name` property. (Access does not support control arrays.) Individual controls (elements) of a control array are designated by their index, starting with 0, up to one less than the number of controls with the same name.

Control menu box See *Application Control menu box* and *Document Control menu box*.

conversation In DDE operations, the collection of Windows messages that are passed between two different applications—the client and server—during an interprocess communication.

correlated subquery A subquery that cannot be independently evaluated. Subqueries depend on an outer query for their result. See also *subquery* and *nested query*.

counter A special field data type of Access 1.x and 2.0 tables that numbers each new record consecutively; called an *AutoNumber field* in Access 95 and 97. See *AutoNumber*.

CUA An abbreviation for Common User Access, an element of IBM's SAA (Systems Application Architecture) specification, which establishes a set of standards for user interaction with menus, dialogs, and other user-interactive portions of an application. The CUA was first implemented in Windows and OS/2 and has been an integral part of these GUIs since their inception.

Oracle Ex. 1009, p. 1179
Oracle v. DeBellis
IPR2024-00221

current database The database opened in Access by choosing File, Open Database (or the equivalent) that contains the objects of an Access application.

current record The record in a Table or Recordset object whose values you modify. The current record supplies values of the current record's data cells to control objects that are bound to the table's fields.

current statement The statement or instruction being executed at a particular instance in time. In debugging or stepwise operation of interpreted applications such as Access, it is the next statement that will be executed by the interpreter when program operation is resumed.

custom control The former name for a control object not native to the application. Access 97 supports 32-bit ActiveX and OLE Controls (OCXs). Visual Basic 3.0 and Visual C++ 3.0 use 16-bit Visual Basic Extension custom controls (VBXs). Visual Basic 4.0 supports 16-bit VBXs and OCXs, plus 32-bit OCXs. See *ActiveX Control* and *OLE Control*.

data access object The container for all of the objects that can be embodied in an Access application, often abbreviated *DAO*. The top member of the data access object hierarchy of Access is the `DBEngine` object, which contains `Workspace`, `User`, and `Group` objects in collections. Database objects are contained in `Workspace` objects. See also *ActiveX Data Object*.

data definition The process of describing databases and database objects such as tables, indexes, views, procedures, rules, default values, triggers, and other characteristics.

data dictionary The result of the data definition process. Also used to describe a set of database system tables that contain the data definitions of database objects, often called *metadata*.

data element The value contained in a data cell, also called a *data item*, or simply an *element*. A piece of data that describes a single property of a data entity, such as a person's first name, last name, Social Security number, age, sex, or hair color. In this case, the person is the data entity.

data entity A distinguishable set of objects that is the subject of a data table and usually has at least one unique data element. A data entity might be a person (unique Social Security number), an invoice (unique invoice number), or a vehicle (unique vehicle ID number, because license plates are not necessarily unique across state lines).

data integrity The maintenance of rules that prevent inadvertent or intentional modifications to the content of a database that would be deleterious to its accuracy or reliability. See *domain integrity* and *referential integrity*.

data modification Changing the content of one or more tables in a database. Data modification includes adding, deleting, or changing information with the `INSERT`, `DELETE`, and `UPDATE` SQL statements. Data modification often is called *updating*.

data sharing The ability to allow more than one user to access information stored in a database from the same or a different application.

Oracle Ex. 1009, p. 1180
Oracle v. DeBellis
IPR2024-00221

data type The description of how the computer is to interpret a particular item of data. Data types are generally divided into two families: strings that usually have text or readable content, and numeric data. The types of numeric data supported vary with the compiler or interpreter used. Most programming languages support a user-defined record or structure data type that can contain multiple data types within it. Field data types, which define the data types of database tables, are distinguished from Access table data types in this book.

database A set of related data tables and other database objects, such as a data dictionary, which are organized as a group.

database administrator The individual(s) responsible for the administrative functions of client-server databases. The database administrator (DBA) has privileges (permissions) for all commands that may be executed by the RDBMS and is ordinarily responsible for maintaining system security, including access by users to the RDBMS itself and performing backup and restoration functions.

database device A file in which databases and related information, such as transaction logs, are stored. Database devices usually have physical names (such as a file name) and a logical name (the parameter of the `USE` statement). In SQL Server, database devices use the DAT file extension.

database object A component of a database. Database objects include tables, views, indexes, procedures, columns, rules, triggers, and defaults. The `DBEngine` object in Access VBA is the topmost member of the class of Access objects. All objects within a single database are subclasses of the `Database` object.

database owner The user who originally created a database. The database owner has control over all of the objects in the database, but may delegate control to other users. Access calls the database owner the *creator*. The database owner is identified by the prefix `dbo` in SQL Server.

Database window The window that appears when you open an Access database and lists the objects (tables, queries, forms, reports, macros, and modules) that are contained in the Database object.

date function A function that provides date and time information or manipulates date and time values.

DCOM An acronym for Distributed Common Object Model that allows communication and manipulation of objects over a network connection. Windows NT 4.0 is the first Microsoft operating system to support DCOM (formerly called NetworkOLE). Microsoft is expected to release a DCOM update for Windows 95 in late 1996. See *COM*.

DDE An abbreviation for dynamic data exchange. DDE is an Interprocess Communication (IPC) method used by Windows and OS/2 to transfer data between different applications. Automation (formerly OLE Automation) provides a more robust IPC method.

Oracle Ex. 1009, p. 1181
Oracle v. DeBellis
IPR2024-00221

deadlock A condition that occurs when two users with a lock on one data item attempt to lock the other's data item. Most RDBMSes detect this condition, prevent its occurrence, and advise both users of the potential deadlock situation.

debug The act of removing errors in the source code for an application.

Debug window A non-modal dialog in which you may enter VBA expressions and view results without writing code in a code editing window. You may also direct information to be displayed in the Debug window by use of the `Debug` object. The appearance of the Debug window varies slightly between VBA-enabled applications.

declaration A statement that creates a user-defined data type, names a variable, creates a symbolic constant, or registers the prototypes of functions incorporated within dynamic link libraries.

declaration section A section of a VBA module reserved for statements containing declarations.

declare In text and not as a keyword, to create a user-defined data type, data holder for a variable or constant. As a VBA keyword, to register a function contained in a dynamic link library in the declarations section of a module.

default A value assigned or an option chosen when no value is specified by the user or assigned by a program statement.

default database The logical name of the database assigned to a user when he or she logs in to the database application.

demand lock Precludes more shared locks from being set on a data resource. Successive requests for shared locks must wait for the demand lock to be cleared.

dependent A condition in which master data in a table (such as invoices) is associated with detail data in a subsidiary table (invoice items). In this case, invoice items are dependent upon invoices.

design-master replica The member of an Access replica set that allows changes in the design of objects, such as tables. The design-master replica usually (but not necessarily) is the .mbd file that is updated by briefcase replicas of the .mbd. See *Briefcase replication*.

Design mode One of three modes of operation of Access, also called *Design view*. Design mode allows you to create and modify tables, queries, forms, reports, and control objects, enter macro actions, and write VBA code. The other two modes are Run mode, also called *run-time* (when the application is executing), and Startup mode (before you open an Access database).

destination document A term used by OLE 1.0 to refer to a compound document.

Oracle Ex. 1009, p. 1182
Oracle v. DeBellis
IPR2024-00221

detail data Data in a subsidiary table that depends on data in a master table to have meaning or intrinsic value. If a user deletes the master invoice records, the subsidiary table's detail data for items included in the invoice lose their reference in the database—they become *orphan data*.

detail table A table that depends on a master table. Detail tables usually have a many-to-one relationship with the master table. See also *detail data*.

device A computer system component that is capable of sending or receiving data, such as a keyboard, display, printer, disk drive, or modem. Windows uses device drivers to connect applications to devices.

device context A Windows term that describes a record (struct) containing a complete definition of all of the variables required to fully describe a window containing a graphic object. These include the dimensions of the graphic area (viewport), drawing tools in use (pen, brush), fonts, colors, Drawing mode, and so on. Windows provides a handle (`hDC`) for each device context.

DHCP Abbreviation for Dynamic Host Configuration Protocol, an Internet standard protocol that allows IP addresses to be pooled and assigned as needed to clients. Windows NT 4.0 includes DHCP Manager, a graphical DHCP configuration tool. See *IP* and *IP address*.

dialog A popup modal child window, also called a *dialog box*, that requests information from the user. Dialogs include message boxes, input boxes, and user-defined dialogs for applications, such as choosing files to open.

DIB An acronym for Device-Independent Bitmap, a Windows-specific bitmap format designed to display graphic information. DIB files take the extension .dip and use a format similar to the .bmp format.

difference In data tables, data elements that are contained in one table but not in another.

directory list An element of a file selection dialog that selectively lists the subfolders of the designated folder of a specified logical drive.

distributed database A database, usually of the client/server type, that is located on more than one database server, often at widely separated locations. Synchronization of data contained in distributed databases is most commonly accomplished by the two-phase commit or replication methods. See *replication* and *two-phase commit*.

disk mirroring Creating on two or more physical disk drives exact duplicates of a disk volume to make files accessible in case of failure of one drive of the mirror set. See *RAID*.

disk striping Distributing the data for a single logical disk volume across two or more physical disk drives. Simple disk striping (RAID 0) provides faster I/O operation. Disk striping with parity (RAID 5) provides faster I/O and protection from failure of a physical disk in a stripe set. See *RAID*.

Oracle Ex. 1009, p. 1183
Oracle v. DeBellis
IPR2024-00221

DLL An abbreviation for dynamic link library, a file containing a collection of Windows functions designed to perform a specific class of operations. Most DLLs carry the .dll extension, but some Windows DLLs, such as Gdi32.exe, use the .exe extension. Functions within DLLs are called (*invoked*) by applications, as necessary, to perform the desired operation.

docfile The file format for creating persistent OLE objects, now called ActiveX documents. Docfiles usually have the extension .ole. Applications that are fully OLE 2-compliant create docfiles with specific extensions, such as .doc (Word) and .xls (Excel). Access 95 .mbd files also are OLE 2 docfiles. OLE 2.1 requires that docfiles include file property values derived from choosing File, Properties. See also *ActiveX documents*.

document A programming object that contains information that originates with the user of the application, rather than being created by the application itself. The data for documents usually is stored in disk files. Access tables, forms, and reports are documents, as are Excel or Lotus 1-2-3 worksheets. In Windows 95 and Windows NT 4.0, a document is a file with an association to an application that can display or manipulate the file.

Document Control menu box The small, square button at the upper-left corner of the menubar of an application that uses the multiple document interface (MDI). Clicking the Document Control menu box displays the Document Control menu. Double-clicking the Document Control menu box closes the document (but not the application). See also *MDI server*.

domain A group of workstations and servers that share a common security account manager (SAM) database and that allow a user to log on to any resource in the domain with a single user ID and password. See also *BDC* and *PDC*. In Access, a domain is a set of records defined by a table or query.

domain aggregate functions A set of functions, identical to the SQL aggregate functions, that you can apply to a specified domain, rather than to one or more Table objects. See also *aggregate functions*.

domain integrity The process of assuring that values added to fields of a table comply with a set of rules for reasonableness and other constraints. As an example, domain integrity is violated if you enter a ship date value that is earlier than an order date. In Access, domain integrity is maintained by field-level and table-level validation rules. See *business rules*.

drag-and-drop A Windows process whereby an icon representing an object, such as a file, can be moved (dragged) by the mouse to another location (such as a different directory) and placed (dropped) in it. You can use drag-and-drop techniques in Access 97's Design mode. Access does not provide the same drag-and-drop capabilities for control objects that are available with Visual Basic.

drive The logical identifier of a disk drive, usually specified as a letter. When used as a component of a path, the drive letter must be followed by a colon and backslash, as in **C:**.

Oracle Ex. 1009, p. 1184
Oracle v. DeBellis
IPR2024-00221

dynamic data exchange See *DDE*.

dynamic link library See *DLL*.

dynaset A set of rows and columns in your computer's memory that represent the values in an attached table, a table with a filter applied, or a query result set. You can update the values of the fields of the underlying table(s) by changing the values of the data cells of an updatable `Dynaset` object. In Access 2+, `Dynaset` is a type of `Recordset` object. See also *recordset*.

embedded object A source document stored as an OLE object in a compound or container document.

empty A condition of a VBA variable that has been declared but has not been assigned a value. Empty is not the same as the `Null` value nor is it equal to the empty or zero-length string (`""`).

enabled The ability of a control object to respond to user actions such as a mouse click, expressed as the `True` or `False` value of the `Enabled` property of the control.

environment A combination of the computer hardware, operating system, and user interface. A complete statement of an environment follows: a 166MHz Pentium computer with a VGA display and two-button mouse, using the Windows 95 operating system.

environmental variable A DOS term for variables that are declared by `PATH` and `SET` statements, usually made in an AUTOEXEC.BAT file, and stored in a reserved memory location by DOS. In Windows 95 and Windows NT, required environmental variables are stored in the Registry, although Windows 95 accepts environmental variables in the AUTOEXEC.BAT file for backward compatibility with 16-bit Windows applications. The environmental variables may be used by applications to adjust their operation for compatibility with user-specific hardware elements or folder structures.

equi-join A `JOIN` in which the values in the columns being joined are compared for equality and all columns in both tables are displayed. This results in two identical columns in the result.

error trapping A procedure by which errors generated during the execution of an application are rerouted to a designated group of lines of code (called an *error handler*) that performs a predefined operation, such as ignoring the error. If errors are not trapped in VBA, the standard modal message dialog with the text message for the error that occurred appears.

event The occurrence of an action taken by the user and recognized by one of Access's event properties, such as `On Click` or `On DblClick`, corresponding to VBA's `Click` and `DblClick` event handlers. Events are usually related to mouse movements and keyboard actions; however, events also can be generated by code using the Timer control object, for example.

Oracle Ex. 1009, p. 1185
Oracle v. DeBellis
IPR2024-00221

event-driven The property of an operating system or environment, such as Windows, that implies the existence of an idle loop. When an event occurs, the idle loop is exited and event-handler code, specific to the event, is executed. After the event handler has completed its operation, execution returns to the idle loop, awaiting the next event.

exclusive lock A lock that prevents others from locking data items until the exclusive lock is cleared. Exclusive locks are placed on data items by update operations, such as SQL's `INSERT`, `UPDATE`, and `DELETE`. In Access and SQL Server, page locking is used; SQL Server 6.5 provides row locking for `INSERT` operations.

executable Code, usually in the form of a disk file, that can be run by the operating system in use to perform a particular set of functions. Executable files in Windows carry the extension .exe and may obtain assistance from dynamic link libraries (DLLs) in performing their tasks.

exponent The second element of a number expressed in scientific notation, the power of 10 by which the first element, the *mantissa*, is multiplied to obtain the actual number. For +1.23E3, the exponent is 3, so you multiply 1.23 by 1,000 (10 to the third power) to obtain the result, 1,230.

expression A combination of variable names, values, functions, and operators that return a result, usually assigned to a variable name. `Result = 1 + 1` is an expression that returns `2` to the variable named `Result`. `DiffVar = LargeVar SmallVar` returns the difference between the two variables to `DiffVar`. Functions may be used in expressions, and the expression may return the value determined by the function to the same variable as that of the argument. `strVar = Mid$(strVar, 2, 3)` replaces the value of `strVar` with three of its characters, starting at the second character.

failover A fault-tolerant clustering architecture in which two servers share a common set of fault-tolerant fixed disk drives. In the event of failure of one of the servers, the other transparently assumes all server processing operations. See *clustering* and *fault tolerance*.

FAT An acronym for file allocation table, the disk file system used by MS-DOS, Windows 95, and (optionally) Windows NT. Windows NT is compatible with the 16-bit FAT system, but not the optional 32-bit FAT (FAT32) for Windows 95 that Microsoft announced in mid-1996. See *HPFS* and *NTFS*.

fault tolerance A computer system's capability to maintain operability, despite failure of a major hardware component such as a power supply, microprocessor, or fixed-disk drive. Fault tolerance requires redundant hardware and modifications to the operating system. Windows NT Server includes fault tolerance for a failed disk drive by disk mirroring (RAID 1) or disk striping with parity (RAID 5). Clustering provides fault tolerance for individual computers. See *clustering* and *RAID*.

fiber A lightweight thread, introduced in Windows NT 4.0, that makes it easier for developers to optimize scheduling within multithreaded applications. See *thread*.

Oracle Ex. 1009, p. 1186
Oracle v. DeBellis
IPR2024-00221

field Synonym for a column that contains attribute values. Also, a single item of information in a record or row.

fifth normal form The rule for relational databases that requires that a table that has been divided into multiple tables must be capable of being reconstructed to its exact original structure by one or more `JOIN` statements.

file The logical equivalent of a table. In dBASE, for instance, each table is a single .dbf file.

file moniker The location of the well-formed path to a persistent OLE 2+ object.

first normal form The rule for relational databases that dictates that tables must be flat. Flat tables can contain only one data value set per row. Members of the data value set, called *data cells*, are contained in one column of the row, and must have only one value.

flag A variable, usually `Boolean` (True/False), that is used to determine the status of a particular condition within an application. The term set is often used to indicate turning a flag from `False` to `True`, and reset for the reverse.

flow control In general usage, conditional expressions that control the sequence of execution of instructions or statements in the source code of an application. `If... Then...End If` is a flow control statement. The term is also used to describe diagrams that describe the mode of operation of an application.

focus A Windows term indicating the currently selected application, or one of its windows, to which all user-generated input (keyboard and mouse operations) is directed. The object with the focus is said to be the *active object*. The title bar of a window with the focus is colored blue for the default Windows color scheme.

font A typeface in a single size, usually expressed in points, of a single style or having a common set of attributes. Font often is misused to indicate a typeface family or style.

foreground In multitasking operations, the application or procedure that is visible on-screen and to which user-generated input is directed. In Windows, the application that has the focus is in the foreground.

foreign key A column or combination of columns whose value must match a primary key in another table when joined with it. Foreign keys need not be unique for each record or row. See also *primary key*.

form A synonym for a user-defined MDI child window in Access. A `Form` object contains the control objects that appear on its surface and the code associated with the events, methods, and properties applicable to the form and its control objects.

form-level Variables that are declared in the Declarations section of an Access form. These variables are said to have *form-level scope*, and are not visible to procedures outside the `Form` object in which the variables are declared, unless declared with the `Public` reserved word.

Oracle Ex. 1009, p. 1187
Oracle v. DeBellis
IPR2024-00221

fourth normal form The rule for relational databases that requires that only related data entities be included in a single table and that tables may not contain data related to more than one data entity when many-to-one relationships exist among the entities.

frame In Windows, a rectangle, usually with a single-pixel-wide border, that encloses a group of objects, usually of the dialog class. When referring to SMPTE timing with MIDI files, it is one image of a motion picture film (1/24 seconds) or one complete occurrence of a television image (approximately 1/30 seconds in NTSC, 1/25 seconds in PAL).

front-end When used in conjunction with database management systems, an application, window, or set of windows by which the user may access and view database records, as well as add to or edit them.

full server An OLE 2-compliant executable application capable of providing embeddable or linked documents for insertion into OLE 2+ container documents. Excel 95, Word 95, Project 4.1, and Wordpad are examples of OLE 2.1 full server applications. Access 97 is not a full server, because you cannot embed or link an Access .mdb file in an OLE 2.1 container application.

function A subprogram called from within an expression in which a value is computed and returned to the program that called it through its name. Functions are classified as internal to the application language when their names are keywords. You may create your own user-defined functions in VBA by adding code between `Function` *`FunctionName`*`...End Function` statements.

global Pertaining to the program as a whole. Global variables and constants are accessible to, and global variables may be modified by, code at the form, module, and procedure level. VBA uses the reserved word `Public` to create or refer to global variables.

global module A code module (container) in which all global variables and constants are declared and in which the prototypes of any external functions contained in DLLs are declared. Use of a global module in Access applications is common, but not required.

grid A preset group of visible or imaginary vertical and horizontal lines used to assist in aligning the position of graphic objects. In Access, the intersection of the imaginary lines is shown as dots on forms and reports in Design mode. Control objects automatically align their outlines to these dots if the Snap To Grid option is enabled. In Access Datasheet view, a set of lines that establish the demarcation of columns and rows.

group In reports, one or more records that are collected into a single category, usually for the purpose of totaling. Database security systems use the term *group* to identify a collection of database users with common permissions. See also *permissions*.

HAL An acronym for Hardware Abstraction Layer, a Windows NT DLL that links specific computer hardware implementations with the Windows NT kernel. Windows NT 4.0 includes HALs for 80x86, Alpha, MIPS, and PowerPC hardware platforms.

handle An unsigned `Long` integer assigned by Windows 95 and Windows NT to uniquely identify an instance (occurrence) of a module (application, `hModule`), task

Oracle Ex. 1009, p. 1188
Oracle v. DeBellis
IPR2024-00221

(hTask), window (hWnd), or device context (hDC) of a graphic object. Handles in 32-bit Windows applications, including applications for Windows 95 and Windows NT, are 32-bit unsigned long integers (dw or double-words). Also used to identify the sizing elements of control objects in Design mode. See also *sizing handle*.

header file A file type used by C and C++ programs to assign data types and names to variables and to declare prototypes of the functions used in the application. C header files usually carry the extension .h.

hierarchical menu A menu with multiple levels, consisting of a main menubar that leads to one or more levels of submenus from which choices of actions are made. Almost all Windows applications use hierarchical menu structures.

host Any computer on a network using the Internet Protocol (IP). See *IP* and *IP address*.

hotlink A term used to describe a DDE (dynamic data exchange) operation in which a change in the source of the DDE data (the server) is immediately reflected in the object of the destination application (the client) which has requested it.

HPFS An abbreviation for the High-Performance File System used by OS/2 and (optionally) Windows NT 3.x. Windows NT 4.0 doesn't support HPFS but can connect via a network to files on HPFS volumes of Windows NT 3.x PCs.

HTML An abbreviation for HyperText Markup Language, a variant of SGML (Standardized General Markup Language), a page-description language for creating files that can be formatted and displayed by World Wide Web browsers.

icon A 32×32-pixel graphic image used to identify the application in the program manager window when the application is minimized, and in other locations in the application chosen by the programmer (such as the Help About dialog). Windows 95 also uses 16×16-pixel icons to identify the application in the title bar.

identifier A synonym for *name* or *symbol*, usually applied to variable and constant names.

idle In Windows, the condition or state in which both Windows and the application have processed all pending messages in the queue from user- or hardware-initiated events and are waiting for the next to occur. The idle state is entered in VBA when the interpreter reaches the `End Sub` statement of the outermost nesting level of procedures for a form or control object.

immediate window Replaced in Access 95 and other VBA-enabled applications by the Debug window. See *Debug window*.

in-place activation The ability to activate an object (launch another application) and have the container application take on the capabilities of the other application. The primary feature of in-place activation (also called *in-situ activation*) is that the other application's menu choices merge with or replace the container application's menu choices in the active window.

Oracle Ex. 1009, p. 1189
Oracle v. DeBellis
IPR2024-00221

in-process A term applied to (OLE) Automation servers, also called *OLE DLLs*, that operate within the same process space (memory allocation) of the (OLE) Automation client using the server. In-process servers commonly are called *InProc servers*. See *out-of-process*.

index For arrays, the position of the particular element with respect to others, usually beginning with 0 as the first element. When used in conjunction with database files or tables, index refers to a lookup table, usually in the form of a file or component of a file, that relates the value of a field in the indexed file to its record or page number and location in the page (if pages are used).

infinite loop A `Do While...Loop`, `For...Next`, or similar program flow control structure in which the condition to exit the loop and continue with succeeding statements is never fulfilled. In `For...Next` loops, infinite looping occurs when the loop counter is set to a value less than that assigned to the `To` embedded keyword within the structure.

initialize In programming, setting all variables to their default values and resetting the point of execution to the first executable line of code. Initialization is accomplished automatically in VBA when you start an application.

inner query Synonym for subquery. See *subquery*.

insertion point The position of the cursor within a block of text. When the cursor is in a text field, it is called the *caret* in Windows.

instance A term used by Windows to describe the temporal existence of a loaded application or one or more of its windows.

instantiate The process of creating an instance of an object in memory.

integer A whole number. In most programming languages, an integer is a data type that occupies two bytes (16 bits). Integers may have signs (as in the VBA `Integer` data type), taking on values from–32,768 to +32,767, or be unsigned. In the latter case, integers can represent numbers up to 65,535.

interface A noun describing a connection between two dissimilar COM objects or (OLE) Automation clients and servers. Another common phrase is *user interface*, meaning the "connection" between the display-keyboard combination and the user. Adapter cards constitute the interface between the PC data bus and peripheral devices such as displays, modems, CD-ROMs, and the like. Drivers act as a software interface between Windows and the adapter cards. A *bridge* is an interface between two dissimilar networks. (OLE) Automation uses `Iole_` interfaces for inter-process communication. Use of *interface* as a verb is jargon.

intersection The group of data elements that are included in both tables that participate in a `JOIN` operation.

invocation path The route through which an object or routine is invoked. If the routine is deeply nested, the path may be quite circuitous.

Oracle Ex. 1009, p. 1190
Oracle v. DeBellis
IPR2024-00221

intranet A private network that uses Internet protocols and common Internet applications (such as Web browsers) to emulate the public Internet. Intranets on LANs and high-speed WANs provide increased privacy and improved performance compared with today's Internet.

invoke To cause execution of a block of code, particularly a procedure or subprocedure. Also used to indicate application of a method to an object.

IP An abbreviation for Internet Protocol, the basic network transmission protocol of the Internet.

IP address The 32-bit hexadecimal address of a host, gateway, or router on an IP network. For convenience, IP addresses are specified as the decimal value of the four address bytes, separated by periods, as in **124.33.15.1**. Addresses are classified as types A, B, and C, depending on the subnet mask applied. See *subnet mask*.

IPX/SPX Abbreviation for Internetwork Packet Exchange/Sequenced Packet Exchange, the transport protocol of Novell NetWare, supported by Windows NT's NWLink service. See *NWLink*.

ISDN An abbreviation for *Integrated Services Digital Network*, a switched telephone service that provides mid-band digital communication capabilities used for Internet connections and for remote access to LANs, as well as voice communication. Windows NT 4.0 has built-in support for ISDN modems, more properly called *network terminators*.

item The name given to the elements contained in a list or the list component of a combo box.

Jet The name given by Microsoft to the database engine native to Access and Visual Basic. The name *Jet* came from an acronym for Joint Engine Technology, the predecessor of Jet 3.5 used by Access and Visual Basic 5.0.

join A basic operation, initiated by the SQL `JOIN` statement, that links the rows or records of two or more tables by one or more columns in each table.

jump In programming, execution of code in a sequence that is not the same as the sequence in which the code appears in the source code. In most cases, a jump skips over a number of lines of code, the result of evaluation of a conditional expression. In some cases, a jump causes another subroutine to be executed.

key or key field A field that identifies a record by its value. Tables are usually indexed on key fields. For a field to be a key field, each data item in the field must possess a unique value. See also *primary key* and *foreign key*.

key value A value of a key field included in an index.

keyword A word that has specific meaning to the interpreter or compiler in use and causes predefined events to occur when encountered in source code. Keywords differ from reserved words because you can use keywords as variable, procedure, or function names. Using keywords for this purpose, however, is not a good programming practice. You cannot use a reserved word as a variable or constant name.

Oracle Ex. 1009, p. 1191
Oracle v. DeBellis
IPR2024-00221

label In VBA programming, a name given to a target line in the source code at which execution results upon the prior execution of a `GoTo LabelName` instruction. A label also is an Access control object that displays, but cannot update, text values.

LAN An acronym for local area network. A LAN is a system comprising multiple computers that are physically interconnected through network adapter cards and cabling. LANs allow one computer to share specified resources, such as disk drives, printers, and modems, with other computers on the LAN.

launch To start a Windows application.

leaf level The lowest level of an index. Indexes are "botmorphic" and derive the names of their elements from the objects found on trees, such as trunks, limbs, and leaves.

library A collection of functions, compiled as a group and accessible to applications by calling the function name, together with any required arguments. DLLs are one type of library; those used by compilers to provide built-in functions are another type.

library database An Access database that is automatically attached to Access when you launch it. Access library databases usually have the extension .mda; encrypted libraries use the extension .mde. Attachment of library databases to Access is controlled by entries in the Registry.

linked object A source document in a compound document that is included by reference to a file that contains the object's data, rather than by embedding the source document in the compound document.

linked table A table that is not stored in the currently open Access database (native or base table), but which you can manipulate as if the table were a native table. Linked tables were called *attached tables* in Access 1.x and 2.0.

list A Windows control object that provides a list of items that the user can choose from with the mouse or the cursor keys.

livelock A request for an exclusive lock on a data item that is repeatedly denied because of shared locks imposed by other users.

local The scope of a variable declared within a procedure, rather than at the form, module, or global level. Local variables are visible (defined) only within the procedure in which they were declared. VBA uses the prefix `Private` to define functions, subprocedures, and variable of local scope.

local area network See *LAN*.

lock A restriction of access to a table, portion of a table, or data item imposed to maintain data integrity of a database. Locks may be *shared*, in which case more than one user can access the locked element(s), or *exclusive*, where the user with the exclusive lock prevents other users from creating simultaneous shared or exclusive locks on the element(s). Access uses *page locks* (2K of the .mdb file), which may lock several adjacent records. Some RDBMSs provide *row locks* that only lock a single record. SQL Server 6.5

Oracle Ex. 1009, p. 1192
Oracle v. DeBellis
IPR2024-00221

uses row locking for `INSERT` operations and page locking for `UPDATE` and `DELETE` operations.

logical A synonym for Boolean. Logical is a data type that may have true or false values only. Logical is also used to define a class of operators whose result is only `True` or `False`. VBA includes a `Boolean` data type.

loop A compound program flow control structure that causes statements contained between the instructions that designate the beginning and end of the structure to be repeatedly executed until a given condition is satisfied. When the condition is satisfied, program execution continues at the source code line after the loop termination statement.

LRPC An acronym for lightweight remote procedure call used for OLE 2+ and some ActiveX operations between OLE clients and OLE full servers on a single computer. LRPC requires that both applications involved in the procedure call be resident on the same computer. See *RPC*.

machine language Program code in the form of instructions that have meaning to and can be acted upon by the computer hardware and operating system employed. Object files compiled from source code are in machine language, as are executable files that consist of object files linked with library files.

macro A set of one or more instructions, called *actions* by Access, that respond to events. Macros and VBA code, which can substitute for Access macros, are used to automate Access applications. Although Access 97 supports macros for backward compatibility with earlier versions, VBA is the preferred method of programming responses to events. Future version of Access may not support macros.

mantissa The first element of a number expressed in scientific notation that is multiplied by the power of 10 given in the exponent to obtain the actual number. For +1.23E3, the exponent is 3, so you multiply the mantissa, 1.23, by 1,000 (10 to the third power) to obtain the result: 1,230.

MAPI Acronym for the Windows Messaging API created by Microsoft for use with Microsoft Mail, which implements Simple MAPI. Microsoft Exchange Server implements MAPI 1.0 (also called *Extended MAPI*).

master database A database that controls user access to other databases, usually in a client-server system.

master table A table containing data on which detail data in another table is dependent. Master tables have a primary key that is matched to a foreign key in a detail table. Master tables often have a one-to-many relationship with detail tables. They sometimes are called *base tables*.

MDI server An OLE 2+ server that supports multiple compound documents within a single running instance of the application.

Oracle Ex. 1009, p. 1193
Oracle v. DeBellis
IPR2024-00221

memo An Access field data type that can store text with a length of up to about 64,000 bytes. (The length of the `Text` field data type is limited to 255 bytes.)

menu A set of choices from which the user determines the next set action to take. The design of menus in Windows is governed by the CUA or Common User Access specification developed by IBM.

metafile A type of graphics file, used by Windows and other applications, that stores the objects displayed in the form of mathematical descriptions of lines and surfaces. Windows metafiles, which use the extension .wmf, are a special form of metafiles. Windows 95 and Windows NT 4.0 also supports enhanced metafiles (EMF).

method One of the characteristics of an object and a classification of keywords in VBA. Methods are the procedures that are applicable to an Access object. Methods that are applicable to a class of objects are inherited by other objects of the same class and may be modified to suit the requirements of the object by a characteristic of an object, called *polymorphism*.

mini-server An applet with OLE server capabilities that you cannot run as a stand-alone application.

mirroring See *disk mirroring*.

MISF An abbreviation for *Microsoft Internet Security Framework*, a set of high-level security services that rely on CryptoAPI 2.0 functions to provide certificate- and password-based authentication. MISF also incorporates secure channel communication using SSL (Secure Sockets Layer) 2.0 and 3.0, plus PCT (Personal Communications Technology), SET (Secure Electronic Transactions) for credit-card purchases, and the Microsoft Certificate Server for issuing authentication certificates.

mission-critical A cliché used in software and hardware advertising to describe the necessity of use of the promoted product if one wants to create a reliable database system.

modal A dialog that must be closed before further action can be taken by the user within the application.

modeless A window or dialog that may be closed or minimized by the user without taking any other action; the opposite of *modal*.

module A block of code, consisting of one or more procedures, for which the source code is stored in a single location (a `Form` or `Module` object in Access). In a compiled language, a code module is compiled to a single object file.

module level Variables and constants that are declared in the `Declarations` section of a module. These variables have module-level scope and are visible (defined) to all procedures that are contained within the module, unless declared `Public`, in which case the variables are visible to all procedures.

moniker A handle to the source of a compound document object.

Oracle Ex. 1009, p. 1194
Oracle v. DeBellis
IPR2024-00221

monitor A name often used in place of the more proper terms, display or video display unit (VDU).

multimedia The combination of sound and graphic images within a single application for the purpose of selling new computer hardware and software. Related outcomes are the creation of animated presentations that incorporate sound effects and graphics, as well as expansion of the market for PCs in the music industry.

multiprocessing The ability of a computer with two or more CPUs to allocate tasks (threads) to a specific CPU. Symmetrical multitasking (SMP), implemented in Windows NT, distributes tasks among CPUs using a load-sharing methodology. Applications must be multithreaded to take advantage of SMP.

multitasking The ability of a computer with a single CPU to simulate the processing of more than one task at a time. Multitasking is effective when one or more of the applications spends most of its time in an idle state waiting for a user-initiated event, such as a keystroke or mouse click.

multithreaded An application that contains more than one thread of execution; a task or set of tasks that executes semi-independently of other task(s). The Jet 3.0 database engine is multithreaded (three threads); Access 95 and VBA are each single-threaded. See *thread*.

multiuser Concurrent use of a single computer by more than one user, usually through the use of remote terminals. UNIX is inherently a multiuser operating system. Access uses the term *multiuser* to refer to Access applications that share a common .mdb file on a network file server.

named pipes A method of interprocess communication, originally developed for OS/2, that provides a secure channel for network communication.

natural join A SQL `JOIN` operation in which the values of the columns engaged in the join are compared, with all columns of each table in the join that do not duplicate other columns being included in the result. Same as an equi-join except that the joined columns are not duplicated in the result.

NBF An abbreviation for NetBEUI Frame, the transport packet structure used by NetBEUI.

nested An expression applied to procedures that call other procedures within an application. The called procedures are said to be *nested* within the calling procedure. When many calls to subprocedures and sub-subprocedures are made, the last one in the sequence is said to be *deeply nested*.

nested object An OLE 2+ compound document incorporated in another OLE 2+ compound document. You can nest OLE 2+ documents as deeply as you like. OLE 1.0 does not supported nested objects.

nested query A SQL `SELECT` statement that contains subqueries. See *subquery*.

Oracle Ex. 1009, p. 1195
Oracle v. DeBellis
IPR2024-00221

NetBEUI An acronym for NetBIOS Extended User Interface, the transport protocol of Microsoft Networking. NetBEUI isn't a routable network, so its popularity is declining in comparison with TCP/IP.

NetBIOS An acronym for Network Basic Input/Output System, the original network API for MS-DOS and the foundation for NetBEUI.

newline pair A combination of a carriage return, the Enter key (`CR` or `Chr$(13)`), and line feed (LF or `Chr$(10)`) used to terminate a line of text on-screen or within a text file. Other characters or combinations may be substituted for the CR/LF pair to indicate the type of newline character (soft, hard, deletable, and so on). The VBA newline constant is `VbCrLf`.

NFS An abbreviation for Network File Server; a file format and set of drivers, created by Sun Microsystems, that allows DOS/Windows and UNIX applications to share a single server disk drive running under UNIX.

non-clustered index An index that stores key values and pointers to data based on these values. In this case, the leaf level points to data pages rather than to the data itself, as is the case for a clustered index. Equivalent to `SET INDEX TO field_name in xBase`.

normal forms A set of five rules, the first three of which originally were defined by Dr. E.F. Cobb, that are used to design relational databases. Five normal forms are generally accepted in the creation of relational databases. See *first normal form*, *second normal form*, *third normal form*, *fourth normal form*, and *fifth normal form*.

normalization Creation of a database according to the five generally accepted rules of normal forms. See also *normal forms*.

not-equal join A `JOIN` statement that specifies that the columns engaged in the join do not equal one another. In Access, you must specify a not-equal join using the SQL `WHERE field1 <> field2` clause.

NT An abbreviation for New Technology used by Windows NT.

NTFS An abbreviation for New Technology File System; Windows NT's replacement for the DOS FAT (file allocation table) and OS/2's HPFS (high-performance file system). NTFS offers many advantages over other file systems, including improved security and the ability to reconstruct files in the event of hardware failures. Windows 3.1+ and Windows 95 can access files stored on NTFS volumes via a network connection, but cannot open NTFS files directly.

null A variable of no value or of unknown value. The default values, `0` for numeric variables and an empty string (`""`) for string variables, are not the same as the `Null` value. The `NULL` value in SQL statements specifies a data cell with no value assigned to the cell.

object In programming, elements that combine data (properties) and behavior (methods) in a single container of code called an *object*. An Access `Form` or `Report` object is a member of the class of Access `Database` objects; a particular control object is a subclass of the control objects class. Objects inherit their properties and methods from the classes

Oracle Ex. 1009, p. 1196
Oracle v. DeBellis
IPR2024-00221

above them in the hierarchy and can modify the properties and methods to suit their own purposes. The code container may be part of the language itself, or you may define your own objects in source code.

object code Code in machine-readable form that can be executed by your computer's CPU and operating system, usually linked with libraries to create an executable file.

object frame An Access control object that contains and displays or plays an OLE object. Bound object frames display or play OLE objects contained in OLE `Object` fields of Access tables. Unbound object frames display or play objects that are either embedded in a `Form` or `Report` object or are linked to a file that supplies the object's data. ActiveX Controls (formerly OLE Controls) are inserted into bound or unbound object frames, depending upon whether or not the ActiveX Control is classified as a data-bound control.

object library A file with the extension .olb that contains information on the objects, properties, and methods exposed by an .exe or .dll file of the same file name that supports OLE Automation.

object permissions Permissions granted by the database administrator for others to view and modify the values of database objects, including data in tables. See also *statement permissions*.

ODBC An abbreviation for the Microsoft Open Database Connectivity API, a set of functions that provides access to client-server RDBMSs, desktop database files, text files, and Excel worksheet files through ODBC drivers. Access 95 uses 32-bit ODBC 3.0 and requires 32-bit ODBC drivers. ODBC most commonly is used to connect to client-server databases, such as Microsoft SQL Server, Sybase System 10+, Informix, and Oracle 7. Access 95 includes a 32-bit ODBC driver for Microsoft SQL Server 4.2+.

ODBCDirect A new feature of the Jet 3.5 database engine that lets you use ODBC to access client/server databases without the necessity to load all of Jet 3.5. ODBCDirect conserves client resources if you only need to connect to SQL Server or another client/server RDBMS. You create an ODBCDirect workspace by using the `dbUseODBC` constant as the value of the *`Type`* argument of the `CreateWorkspace` method. ODBCDirect is closely related to the Remote Data Object (RDO). See also *RDO*.

offset The number of bytes from a reference point, usually the beginning of a file, to the particular byte of interest. The first byte in a file, when offset is used to specify location, is always 0.

OLE Automation An extension of OLE 2+ that provides the framework (interfaces) for applications and libraries to expose programmable objects that can be manipulated by client applications. Applications that expose programmable objects are called (OLE) Automation servers. See *automation* and *programmable object*.

OLE Control An in-process OLE Automation server with the extension .ocx that exposes a single object, plus the properties, methods, and events of the object. Exposing events differentiates OLE Controls from other types of (OLE) Automation servers. Access

Oracle Ex. 1009, p. 1197
Oracle v. DeBellis
IPR2024-00221

95 referred to OLE Controls (the first Microsoft term) as *Custom Controls*; the term *Custom Control* is more commonly used for 16-bit Visual Basic Extensions, VBXs. OLE Controls have been superseded by 32-bit ActiveX Controls. *See ActiveX Controls.*

OLE DB A new Microsoft framework for providing a uniform interface to data from a variety of sources, including text files and mainframe databases. OLE DB doesn't replace ODBC, but includes an ODBC provider that takes the place of the ODBC driver manager. See also *ActiveX Data Objects* and *ODBC.*

OLE DLL A synonym for an in-process OLE Automation server implemented as a Windows DLL. See *in-process.*

OpenDoc A standard proposed by Apple Computer, Borland International, Lotus Development, Novell, and other competitors of Microsoft to supplant or replace OLE 2+.

operand One of the variables or constants upon which an operator acts. In 1 + 2 = 3, both 1 and 2 are operands; + and = are the operators. See *operator.*

operating system Applications that translate basic instructions, such as keyboard input, to language understood by the computer. The most common operating systems used with personal computers are MS-DOS (Microsoft Disk Operating System), Windows 95, Windows NT, UNIX, and OS/2.

operator A keyword or reserved symbol that, in its unary form, acts on a single variable, or otherwise acts on two variables, to give a result. Operators may be of the conventional mathematics type such as + (add), – (subtract), / (divide), and * (multiply), as well as logical, such as `And` or `Not`. The unary minus (–), when applied to a single variable in a statement such as `intVar = - intVar`, inverts the sign of `intVar` from – to + or from + to –.

optimistic locking A method of locking a record or page of a table that makes the assumption that the probability of other users locking the same record or page is low. With optimistic locking, the record or page is locked only when the data is updated, not during the editing process (`LockEdits` property set to `False`).

option button A synonym for *radio button*, the original terminology in the CUA specification. Option buttons are circular control objects whose center is filled when selected. If grouped, only one option button of a group can be selected. If placed directly on a form, the form becomes the group and only one option button can be selected.

outer join An SQL `JOIN` operation in which all rows of the joined tables are returned, whether or not a match is made between columns. SQL database managers that do not support the `OUTER JOIN` reserved words use the `*=` (`LEFT JOIN`) operator to specify that all of the rows in the preceding table return, and `=*` (`RIGHT JOIN`) to return all of the rows in the succeeding table.

outer query A synonym for the primary query in a statement that includes a subquery. See also *subquery.*

out-of-process An (OLE) Automation server in the form of an executable (.exe) file that operates in its own process space (memory allocation) and uses LRPCs (lightweight

Oracle Ex. 1009, p. 1198
Oracle v. DeBellis
IPR2024-00221

remote procedure calls) to communicate with the Automation client. The term *OutOfProc* often is used as shorthand for *out-of-process*.

page In tables of client-server RDBMSs, such as Microsoft SQL Server and Access databases, a 2K block that contains records of tables. Client-server and Access databases lock pages, while DOS desktop databases usually lock individual records. Page-locking is required by most RDBMSs when variable-length records are used in tables.

parameter The equivalent of an argument, but associated with the procedure that receives the value of an argument from the calling function. The terms *parameter* and *argument*, however, are often used interchangeably.

parse The process of determining if a particular expression is contained within another expression. Parsing breaks program statements into keywords, operators, operands, arguments, and parameters for subsequent processing of each by the computer. Parsing string variables involves searching for the occurrence of a particular character or set of characters in the string, and then taking a specified set of actions when found or not found.

PDC An abbreviation for Primary Domain Controller, the Windows NT server in a domain that's responsible for maintaining user and group accounts for a domain. Primary and Backup Domain Controllers authenticate domain users during the logon process. See *BDC*.

permissions Authority given by the system administrator, database administrator, or database owner to perform operations on a network or upon data objects in a database.

persistent (graphics) A Windows graphic image that survives movement, resizing, or overwriting of the window in which it appears. Persistent images are stored in global memory blocks and are not released until the window containing them is destroyed.

persistent (objects) An object that is stored in the form of a file or an element of a file, rather than only in memory. `Table` and `QueryDef` objects are persistent because these objects are stored in .mdb files. `Recordset` objects, on the other hand, are stored in memory. Such objects are called *temporal* or *impersistent objects*.

pessimistic locking A method of locking a record or page of a table that makes the assumption that the probability of other users locking the same record or page is high. With pessimistic locking, the record or page is locked during the editing and updating process (`LockEdits` property set to `True`).

point In typography, the unit of measurement of the vertical dimension of a font, about 1/72 of an inch. The point is also a unit of measurement in Windows, where it represents exactly 1/72 of a logical inch or 20 twips. Unless otherwise specified, all distance measurements in VBA are in twips.

pointer A data type that comprises a number representing a memory location. Near pointers are constrained to the 64K default local data segment. Far pointers can access any location in the computer's memory. Pointers are used extensively in C-language applications to access elements of arrays, strings, structures, and the like. VBA has only

Oracle Ex. 1009, p. 1199
Oracle v. DeBellis
IPR2024-00221

one pointer data type—to a zero-terminated string when the `ByVal...As String` keywords are applied to a VBA string passed to an external function contained in a dynamic link library.

poke In DDE terminology, the transmission of an unrequested data item to a DDE server by the DDE client. In BASIC language terminology, placing a byte of data in a specific memory location. VBA does not support the BASIC `POKE` keyword and uses the `DDEPoke` method for DDE operations.

PPP An abbreviation for Point-to-Point Protocol, the most common Internet protocol for connection to TCP/IP networks via conventional and ISDN modems.

PPTP An abbreviation for Point-to-Point Tunneling Protocol, a Microsoft-sponsored protocol, included with Windows NT 4.0, that uses encryption to assure privacy of communication over the Internet. See *VPN*.

precedence The sequence of execution of operators in statements that contain more than one operator.

primary key The column or columns whose individual or combined values (in the case of a composite primary key) uniquely identify a row in a table.

primary verb The default verb for activating an OLE 2+ object. `Edit` is the default verb for most OLE objects, except multimedia objects, whose default verb is usually `Play`.

print zone The area of a sheet of paper upon which a printer can create an image. For most laser printers and standard dot-matrix printers, this is 8 inches in width. The vertical dimension is unlimited for dot-matrix printers and usually is 13.5 inches for a laser printer with legal-size paper capabilities.

printer object A VBA object representing the printer chosen as the default by the Control Panel Printers function's Set Default choice.

procedure A self-contained collection of source code statements, executable as an entity. All VBA procedures begin either with the reserved word `Sub` or `Function`, which may be preceded by the `Public`, `Private`, or `Static` reserved words, and terminate with `End Sub` or `End Function`.

process server An "unofficial" term used in this book to specify an (OLE) Automation server, either in-process or out-of-process, that does not provide user-interface components, such as dialogs or windows. Process servers often are used in three-tier client-server applications to implement business rules. See *business rules* and *three-tier*.

program All of the code required to create an application, consisting basically of declarations, statements, and in Windows, resource definition and help files.

programmable object An object exposed by an (OLE) Automation server, together with a set of properties and methods applicable to the object. The exposed object can be manipulated by the application programming language of an (OLE) Automation client application.

Oracle Ex. 1009, p. 1200
Oracle v. DeBellis
IPR2024-00221

projection Identifies the desired subset of the columns contained in a table. You create a projection with a query that defines the fields of the table you want to display, but without criteria that limit the records that are displayed.

Properties window A window that displays the names and properties of Access `Table`, `Form`, `Report`, and `Control` objects.

property One of the two principal characteristics of objects (the other is methods). Properties define the manifestation of the object, for example, its appearance. Properties may be defined for an object or for the class of objects to which the particular object belongs, in which case they are said to be *inherited*.

protocol A description of the method by which networked computers communicate. Windows NT and Windows 95 allow the simultaneous use of multiple network protocols, including TCP/IP, NetBEUI, and IPX/SPX.

protocol stack Network protocol software that implements a specific protocol, such as TCP/IP.

pseudo-object Objects that are contained within other OLE 2+ objects, such as the cells of a spreadsheet object.

qualification A search condition that data values must meet to be included in the result of the search.

qualified To precede the name of a database object with the name of the database and the object's owner, or to precede the name of a file with its drive designator and the path to the directory in which the file is stored. The terms *well-qualified path* and *well-formed path* to a file appear often in documentation.

query A request to retrieve data from a database with the SQL `SELECT` instruction or to manipulate data in the database, called an *action query* by Access.

QueryDef A persistent Access object that stores the Access SQL statements that define a query. `QueryDef` objects are optimized, when applicable, by the Jet database engine's query optimizer and stored in a special optimized format.

RAID An acronym for redundant array of inexpensive disks, a method of connecting multiple disk drives to a single controller card to achieve faster data throughput, data storage redundancy for fault tolerance, or both. See *disk mirroring*, *disk striping*, and *fault tolerance*.

RDBMS An abbreviation for relational database management system. An RDBMS is an application that is capable of creating, organizing, and editing databases; displaying data through user-selected views; and printing formatted reports. Most RDBMSs include at least a macro or macro language, and most provide a system programming language. dBASE, Paradox, and FoxPro are desktop RDBMSs.

record A synonym for a user-defined data type, called a *structure* in C and C++. Record also is used in database applications to define a single element of a relational database

Oracle Ex. 1009, p. 1201
Oracle v. DeBellis
IPR2024-00221

file that contains each of the fields defined for the file. Records need not contain data to exist, but Jet does not append a record without a value in at least one field. A record is the logical equivalent of the row of a table. A set of related fields or columns of information that are treated as a unit by an RDBMS application.

recursion A condition in which a procedure or function calls itself. As a general rule, you should avoid recursive procedures and functions in VBA unless you are an experienced programmer.

redirector Software that intercepts requests for remotely provided services, such as files in server shares, and sends the request to the appropriate computer on the network.

reference In VBA, the incorporation of pointers to specific sets of programmable objects exposed by Automation servers and manipulated by VBA code in the Automation client. You create a VBA reference to a set of objects exposed by an Automation server, such as Microsoft Excel 97, in the References dialog that is accessible by choosing Tools, References when a module is the active Access object. Once you declare a reference to the set of objects, the VBA pseudo-compiler checks the syntax of your code against the syntax specified for the referenced object. You also can use predefined intrinsic constants for the referenced objects in your VBA code.

referential integrity Rules governing the relationships between primary keys and foreign keys of tables within a relational database that determine data consistency. Referential integrity requires that the values of every foreign key in every table be matched by the value of a primary key in another table. Access 2+ includes features for maintaining referential integrity, such as cascading updates and cascading deletions.

refresh To redisplay records in Access's datasheet views or in a form or report so as to reflect changes others in a multiuser environment have made to the records.

Registry A database that contains information required for the operation of Windows 95 and Windows NT, plus applications installed under Windows 95 and Windows NT. The Windows Registry takes the place of Windows 3.1+'s REG.DAT, WIN.INI, and SYSTEM.INI files, plus *PROFILE*.INI files installed by Windows 3.1 applications. The Registry also includes user information, such as user IDs, encrypted passwords, and permissions. Windows 95 and Windows NT include Rwgedit.exe for editing the Registry. ActiveX Components and OLE 2+ servers add entries to the Registry to specify the location of their exe files. Automation servers add Registry entries for each of the objects they expose. The Windows NT and Windows 95 registries differ in structure and thus are incompatible.

relation Synonym for a table or a data table in an RDBMS.

relational database See *RDBMS*.

relational operators Consist of operators such as >, <, <>, and = that compare the values of two operands and return `True` or `False` depending on the values compared. They are sometimes called *comparative operators*.

Oracle Ex. 1009, p. 1202
Oracle v. DeBellis
IPR2024-00221

Remote Automation Object An out-of-process (OLE) Automation server, usually called an RAO, that resides on a server and is accessible to RAO-compliant applications that connect to the server with DCOM. See also *DCOM*.

Remote Data Object A substitute for the Jet 3.5 Data Access Object that provides a more direct connection to the ODBC API. Access 97 offers ODBCDirect as an alternative to RDO. You can reference the RDO 2.0 type library in Access 97 applications and write VBA code for the RDO to speed queries against client-server databases. Like ODBCDirect, RDO also offers additional RDBMS connection management features. See also *ODBCDirect*.

remote procedure call (RPC) An interprocess communication method that allows an application to run specific parts of the application on more than one computer in a distributed computing environment. Visual Basic 5.0 is capable of creating Remote Automation Objects (RAOs) that use RPCs for communication over a network.

replication The process of duplicating database objects (usually tables) in more than one location, including a method of periodically rationalizing (synchronizing) updates to the objects. Unlike Access 95, Access 97 supports partial replication. Database replication is an alternative to the two-phase commit process. Microsoft SQL Server 6.5 supports replication of databases across multiple Windows NT servers. Access 97 includes a mini-replication feature designed for mobile users of Access databases running Windows 95. See *Briefcase replication* and *two-phase commit*.

reserved word A word that comprises the vocabulary of a programming language and that is reserved for specific use by the programming language. You cannot assign a reserved word as the name of a constant, variable, function, or subprocedure. Although the terms *reserved word* and *keyword* often are used interchangeably, they do not describe an identical set of words. See also *keyword*.

restriction A query statement that defines a subset of the rows of a table based on the value of one or more of its columns.

RGB A method of specifying colors by using numbers to specify the individual intensities of its red, green, and blue components, the colors created by the three "guns" of the cathode-ray tube (CRT) of a color display.

RIFF An acronym for the Windows Resource Interchange File Format used in conjunction with the Multimedia Extensions to Windows. Depending upon their definition, these files may contain MIDI sequence, sample dump or system exclusive data, waveform files, or data to create graphic images. RIFF is the preferred file format, at least by Microsoft Corporation, for multimedia files.

rollback A term used in transaction processing that cancels a proposed transaction that modifies one or more tables and undoes changes, if any, made by the transaction prior to a `COMMIT` or `COMMIT TRANSACTION` SQL statement.

routine A synonym for *procedure*.

Oracle Ex. 1009, p. 1203
Oracle v. DeBellis
IPR2024-00221

row A set of related columns that describes a specific data entity. A synonym for *record*.

row aggregation functions See *aggregate functions*.

rule A specification that determines the type of data and value of data that may be entered in a column of a table. Rules are classified as validation rules and business rules. See *business rules*.

Run mode The mode of Access operation when Access is executing your database application. Run mode is called *run-time* by Microsoft; however, the term *run-time* normally refers to errors that occur when running the executable version of an application.

running state An OLE 2+ object is in the running state when the application that created the object is launched and has control of the object.

SAM An acronym for Security Accounts Manager, a Windows NT subsystem that maintains a database of user account names and passwords for authentication.

scalable The property of a multiprocessing computer that defines the extent to which addition of more processors increases aggregate computing capability. Windows NT 4.0 Server is generally considered to be scalable to eight Intel processors.

scope In programming terminology, the extent of visibility (definition) of a variable. VBA has global (`Public`, visible to all objects and procedures in the application), form/report (visible to all objects and procedures within a single form or report), module (visible to all procedures in a single module file), and local (`Private`, visible only within the procedure in which declared) scope. The scope of a variable depends upon where it is declared. See also *global*, *form-level*, *module level*, and *local*.

screen object An Access VBA object and object class defined as the entire usable area of the video display unit. All visible form and control objects are members of subclasses of the `Screen` object.

scroll bar Vertical and horizontal bars at the right side and bottom, respectively, of a multiline text box that allow the user to scroll the window to expose otherwise hidden text. Access also provides scroll bars for tables and queries in Run mode (Datasheet view) and for forms or reports that exceed the limits of the display.

SDI server An OLE 2+ server that supports only a single compound document (Single Document Interface) within an instance of the application. SDI is the preferred design of applications for Windows 95; however, all Microsoft Office 95 productivity applications are multiple-document interface (MDI) applications. Windows 95's Explorer and Exchange client are examples of SDI applications.

second normal form The rule for relational databases requiring columns that are not key fields each be related to the key field. That is, a row may not contain values in data cells that do not pertain to the value of the key field. In an invoice item table, for instance, the columns of each row must pertain solely to the value of the invoice number key field.

Oracle Ex. 1009, p. 1204
Oracle v. DeBellis
IPR2024-00221

seek To locate a specific byte, record, or chunk within a disk file. The `Seek` method of Access VBA can only be used in conjunction with Recordset objects of the `Table` type and requires that the table be indexed.

select list The list of column names, separated by commas, that specify the columns to be included in the result of a `SELECT` statement.

selection In Windows, one or more objects that have been chosen by clicking the surface of the object with the mouse or otherwise assigning the focus to the object. When used in conjunction with text, selection means the highlighted text that appears in a text box or window. See also *restriction*.

self-join A SQL `JOIN` operation used to compare values within the columns of one table. Self-joins join a table with itself, requiring that the table be assigned two different names, one of which must be an alias.

separator A reserved symbol used to distinguish one item from another, as exemplified by the use of the exclamation point (`!`, bang character) in Access to separate the name of an object class from a specific object of the class, and an object contained within a specified object. The period separator (`.`, dot) separates the names of objects and their methods or properties.

sequential access file A file in which one record follows another in the sequence applicable to the application. Text files, for the most part, are sequential.

server A computer on a LAN that provides services or resources to client computers by sharing its resources. Servers may be *dedicated*, in which case they share their resources but do not use them themselves except in performing administrative tasks. Servers in client-server databases are ordinarily dedicated to making database resources available to client computers. Servers may also be used to run applications for users, in which case, the server is called an *application server*. Peer-to-peer or workgroup servers, such as servers created by using Windows 95 and Windows NT to share disk folders, are another class of server.

session In Access 97, an instance of the Jet 3.5 database engine for a single user, represented by the `Workspace` object. You can establish multiple sessions which become members of the `Workspaces` collection. In RDBMS terminology, the period between the time that a user opens a connection to a database and the time that the connection to the database is closed.

shared application memory Memory that is allocated between processes involved in an LRPC call. See also *LRPC*.

shared lock A lock created by read-only operations that does not enable the user who creates the shared lock to modify the data. Other users can place shared locks on data so they can read it, but no user may apply an exclusive lock on the data while any shared locks are in effect.

Oracle Ex. 1009, p. 1205
Oracle v. DeBellis
IPR2024-00221

shortcut key A Ctrl+*key* combination that provides access to a menu choice, macro, or other function of the application in lieu of selection with the mouse.

SID An acronym for security ID, a numeric value that identifies a logged-on user who has been authenticated by Windows NT or a user group.

single-stepping A debugging process by which the source code is executed one line at a time to allow you to inspect the value of variables, find infinite loops, or remove other types of bugs.

sizing handle The small black rectangles on the perimeter of Access control objects that appear on the surface of the form or report in Design mode when the object is selected. You drag the handles of the rectangles to shrink or enlarge the size of control objects.

SMB An abbreviation for Server Message Block, a networking protocol used by NetBEUI to implement Microsoft Networking.

source code The readable form of code that you create in a high-level language. Source code is converted to machine-language object code by a compiler or interpreter.

source document A term used by OLE 1.0 to refer to a compound object in a container document.

SQL An acronym, pronounced either as "sequel" or "seekel," for Structured Query Language, a language developed by IBM Corporation for processing data contained in mainframe computer databases. (Sequel is the name of a language, similar to SQL, developed by IBM but no longer in use.) SQL has now been institutionalized by the creation of an ANSI standard for the language.

SQL aggregate functions See *aggregate functions*.

statement A syntactically acceptable (to the interpreter or compiler of the chosen language) combination of instructions or keywords and symbols, constants, and variables that must appear on a single line or use the line continuation pair (a space followed by an underscore) to use multiple lines.

static When applied to a variable, a variable that retains its last value until another is assigned, even though the procedure in which it is defined has completed execution. All global variables are static. Variables declared as `Static` are similar to global variables; however, their visibility is limited to their declared scope. Static is also used to distinguish between statically linked (conventional) executable files and those that use DLLs.

stored procedure A set of SQL statements (and with those RDBMSs that support them, flow control statements) that are stored under a procedure name so that the statements can be executed as a group by the database server. Some RDBMSs, such as Microsoft and Sybase SQL Server, precompile stored procedures so that they execute more rapidly.

Oracle Ex. 1009, p. 1206
Oracle v. DeBellis
IPR2024-00221

string A data type used to contain textual material, such as alphabetic characters and punctuation symbols. Numbers may be included in or constitute the value of string variables, but cannot be manipulated by mathematical operators.

stripe set See *disk striping* and *fault tolerance*.

structure Two or more keywords that are used together to create an instruction, which is usually conditional in nature. In C and C++ programming, a user-defined data type. See also *compound*.

Structured Query Language See *SQL*.

stub A procedure or user-defined function that, in VBA, consists only of `Sub SubName...End Sub` or `Function FnName...End Function` lines with no intervening code. Stubs for subprocedures are created automatically by Access for event-handling code stored in `Form` and `Report` objects. Stubs are used to block out the procedures required by the application that can be called by the Main program. The intervening code statements are filled in during the programming process.

style In typography, a characteristic or set of attributes of a member of a family of typefaces created by an outline or bitmap designed specifically to implement it. Styles include bold, italic, bold-italic, bold-italic-condensed, and so forth. Styles may contain attributes for weight (bold, demi-bold, black), form (italic, roman), and spacing (compressed or extended) in various combinations.

subform A form contained within another form.

submenu A set of choices presented when a main menu choice is made. In Windows, the first-level submenus are similar to drop-down dialogs. Second-level submenus usually appear horizontally at the point of the first submenu choice.

subnet mask A local bit mask (set of flags) that specifies which bits of the IP address specify a particular IP network or a host within a subnetwork. An IP address of 128.66.12.1 with a subnet mask of 255.255.255.0 specifies host 1 on subnet 128.66.12.0. The subnet mask determines the maximum number of hosts on a subnetwork.

subprocedure A procedure called by another procedure other than the main procedure (WinMain in Windows). In Access, all procedures except functions are subprocedures because Msaccess.exe contains the WinMain function.

subquery A SQL `SELECT` statement that is included (nested) within another `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement, or nested within another subquery.

subreport A report contained within another report.

syntax The rules governing the expression of a language. As with English, Spanish, Esperanto, or Swahili, programming languages each have their own syntax. Some languages allow much more latitude (irregular forms) in their syntax. VBA has a relatively rigid syntax, while C provides more flexibility at the expense of complexity.

Oracle Ex. 1009, p. 1207
Oracle v. DeBellis
IPR2024-00221

system administrator The individual(s) responsible for the administrative functions for all applications on a LAN or users of a UNIX cluster or network, usually including supervision of all databases on servers attached to the LAN. If the system administrator's (SA's) responsibility is limited to databases, the term *database administrator* (DBA) is ordinarily assigned.

system colors The 20 standard colors used by Windows for elements of its predefined objects such as backgrounds, scroll bars, borders, and title bars. You may change the system colors from the defaults through Control Panel's Color and Desktop functions.

system databases Databases that control access to databases on a server or across a LAN. Microsoft SQL Server has three system databases: the master database, which controls user databases; tempdb, which holds temporary tables; and model, which is used as the skeleton to create new user databases. Any database that is not a user database is a system database.

system function Functions that return data about the database rather than from the content of the database.

system object An object defined by Access rather than by the user. Examples of system objects are the `Screen` and `Debug` objects.

system table A data dictionary table that maintains information on users of the database manager and each database under the control by the system. Access system tables carry the prefix `MSys`.

T-1 The most common moderate-speed telecommunication connection between LANs to create WAN. Dedicated T-1 lines provide 1.544Mbps of bandwidth. T-1 lines also are the most common method of connecting servers to the Internet.

tab order The order in which the focus is assigned to multiple control objects within a form or dialog with successive pressing of the Tab key.

table A database object consisting of a group of rows (records) divided into columns (fields) that contain data or Null values. A table is treated as a database device or object.

TCP/IP Abbreviation for Transport Control Protocol/Internet Protocol, the networking protocol of the Internet, UNIX networks, and the preferred protocol for Windows NT networks. TCP/IP is a routable network that supports subnetworks. See *IP*.

TDI An abbreviation for Transport Driver Interface, used by Windows NT to implement multiple network protocols by using various network interface cards.

text box A Windows object designed to receive printable characters typed from the keyboard. Access provides two basic types: single and multiline. Entries in single-line text boxes are terminated with an Enter keystroke. Multiline text boxes accept more than one line of text, either by a self-contained word-wrap feature (if a horizontal scroll bar is not present) or by a Ctrl+Enter key combination.

Oracle Ex. 1009, p. 1208
Oracle v. DeBellis
IPR2024-00221

text file A disk file containing characters with values ordinarily ranging from `Chr$(1)` through `Chr$(127)` in which lines of text are separated from one another with newline pairs (`Chr$(13)` & `Chr$(10)`).

theta join A SQL `JOIN` operation that uses comparison or relational operators in the `JOIN` statement. See also *operator*.

third normal form The rule for relational databases that imposes the requirement that a column that is not a key column may not be dependent upon another column that is not a key column. The third normal form is generally considered the most important because it is the first in the series that is not intuitive.

thread A part of a process, such as an executing application, that can run as an object or an entity.

three-tier The architecture of a database application, usually involving a client-server RDBMs, where the front-end application is separated from the back-end RDBMS by a middle tier application. In Access and Visual Basic applications, the middle tier usually is implemented as an OLE Automation process server, which implements the database connection, enforces business rules, and handles transfer of data to and from databases of the RDBMS. See *business rules* and *process server*.

time stamp The date and time data attributes applied to a disk file when created or edited. SQL Server and the ODBC API support the timestamp field, which resolves concurrency issues when updating tables.

timer An Access control object that is invisible in Run mode and that is used to trigger a `Timer` event at preselected intervals.

title bar The heading area of a window, usually blue, in which the title of the window appears, usually in bright white (reverse).

toggle A property of an object, such as a check box, that alternates its state when repeatedly clicked with the mouse or activated by a shortcut key combination.

toolbar A group of command button icons, usually arranged horizontally across the top of a window, that perform functions that would ordinarily require one or more menu choices. *Floating toolbars* can be located anywhere on your display.

toolbox A collection of command buttons designated as tools, usually with icons substituted for the default appearance of a command button, that choose a method applicable to an object (usually graphic) until another tool is selected. An example is the Access Toolbox.

topic In DDE conversations, the name of the file or other identifying title of a collection of data. When used in conjunction with help files, the name of the subject matter of a single help screen display.

Oracle Ex. 1009, p. 1209
Oracle v. DeBellis
IPR2024-00221

TRANSACT-SQL A superset of ANSI SQL used by Microsoft and Sybase SQL Server. TRANSACT-SQL includes flow control instructions and the capability to define and use stored procedures that include conditional execution and looping.

transaction A group of processing steps that are treated as a single activity to perform a desired result. A transaction might entail all of the steps necessary to modify the values in or add records to each table involved when a new invoice is created. RDBMSs that are capable of transaction processing usually include the capability to cancel the transaction by a rollback instruction or to cause it to become a permanent part of the tables with the `COMMIT` or `COMMIT TRANSACTION` statement. See *rollback*.

trigger A stored procedure that occurs when a user executes an instruction that may affect the referential integrity of a database. Triggers usually occur prior to the execution of `INSERT`, `DELETE`, or `UPDATE` statements so that the effect of the statement on referential integrity can be examined by a stored procedure prior to execution. See also *stored procedure*.

trust In Windows NT domain terminology, a relationship between domain controllers in which users who are members of the trusted domain can access services on another trusting domain without the need to log onto the trusting domain.

twip The smallest unit of measurement in Windows and the default unit of measurement of VBA. The twip is 1/20 of a point, or 1/1440 of a logical inch.

two-phase commit A process applicable to updates to multiple (distributed) databases that prevents a transaction from completing until all of the distributed databases acknowledge that the transaction can be completed. The replication process has supplanted two-phase commit in most of today's distributed client-server RDBMSs. See *replication*.

type See *data type*.

type library A file with the extension .tlb that provides information about the types of objects exposed by an ActiveX component or (OLE) Automation server. The type library for Msaaccess.exe is Msaccess.tlb. See *object library*.

typeface Synonym for face. A set of fonts of a single family in any available size possessing an identical style or set of attributes.

unary See *operator*.

UNC An abbreviation for Uniform Naming Convention, the method of identifying the location of files on a remote server. UNC names begin with `\\`. Windows 95 and Windows NT support UNC; 32-bit Windows applications must support UNC to qualify for application of Microsoft's "Designed for Windows 95" logo. All Microsoft Office 95 applications support UNC.

Oracle Ex. 1009, p. 1210
Oracle v. DeBellis
IPR2024-00221

Unicode A replacement for the 7-bit or 8-bit ASCII and ANSI representations of characters with a 16-bit model that allows a wider variety of characters to be used. Windows 95 and Windows NT support Unicode. Access 95 automatically converts Unicode to ANSI and vice-versa.

uniform data transfer (UDT) The interprocess communication (IPC) method used by OLE 2+. OLE 1.0 uses DDE for IPC.

unique index An index in which no two key fields or combinations of key fields upon which the index is created may have the same value.

UNIX Registered trademark of Novell Incorporated (formerly of AT&T) for its multiuser operating system, now administered by the Open Systems Foundation (OSF). Extensions and modifications of UNIX include DEC Ultrix, SCO UNIX, IBM AIX, and similar products.

update A permanent change to data values in one or more data tables. An update occurs when the `INSERT`, `DELETE`, `UPDATE`, or `TRUNCATE TABLE` SQL commands are executed.

user-defined A data type, also called a record, that is specified in your VBA source code by a `Type...End Type` declaration statement in the Declarations section of a module. The elements of the user-defined record type can be any data type valid for the language and may include other user-defined types.

user-defined transaction A group of instructions combined under a single name and executed as a block when the name is invoked in a statement executed by the user.

validation The process of determining if an update to a value in a table's data cell is within a preestablished range or is a member of a set of allowable values. Validation rules establish the range or set of allowable values. Access 2+ supports validation rules at the field and table levels.

variable The name given to a symbol that represents or substitutes for a number (numeric), letter, or combination of letters (string).

VBA An abbreviation for Visual Basic for Applications, the official name of which is "Visual Basic, Applications Edition." VBA is Microsoft's common application programming (macro) language for Access 97, Excel 5+, Project 4+, Word 8, and Visual Basic 4+. Each application has its own "flavor" of VBA as a result of automatically created references to the application's object hierarchy in VBA code. Thus, this book uses the terms *Excel VBA* and *Project VBA* when referring to a particular flavor of VBA. VBA alone is used when the subject matter is applicable to all current flavors of VBA.

view The method by which the data is presented for review by the user, usually on the computer display. Views can be created from subsets of columns from one or more tables by implementing the SQL `CREATE VIEW` instruction.

Visual Basic for Applications See *VBA*.

Oracle Ex. 1009, p. 1211
Oracle v. DeBellis
IPR2024-00221

VM Abbreviation for virtual memory, a method of mapping a combination of RAM and images of RAM stored in a paging file to provide an address space larger than that available from the RAM installed in the computer.

VM manager The Windows NT executive service that loads memory images stored in a paging file on demand, as well as saving memory images in the paging file when no longer needed by a thread.

VPN An abbreviation for Virtual Private Network, a means of establishing secure communication channels on the Internet using various forms of encryption. See *PPTP*.

WAN An acronym for wide area network. A WAN is a system for connecting multiple computers in different geographical locations through the use of the switched telephone network or leased data lines, by optical or other long-distance cabling, or by infrared, radio, or satellite links.

WAVE file A file containing waveform audio data, usually with a .wav extension.

Waveform audio A data type standard of the Windows Multimedia Extensions that defines how digitally sampled sounds are stored in files and processed by Windows API functions calls.

WDM An abbreviation of Windows Driver Model, a 32-bit architecture for creating device drivers that run under both Windows NT and Windows 95.

wild card A character that substitutes for and allows a match by any character or set of characters in its place. The DOS ? and * wild cards are similarly used in Windows applications.

Win32 An API for creating 32-bit applications that run under Windows 95 and Windows NT. Applications that are written to the Win32 API are purported to provide substantially improved performance when run under Windows 95 and Windows NT.

Win32s A subset of the Win32 API designed to add limited 32-bit capabilities to Windows 3.1+. Very few applications have been written to the Win32s API, which appears to have become obsolete.

WinHelp32 A contraction used to describe the Windows help engine of Windows 95 and the files that are used in the creation of Windows 95 help systems. WinHelp 32 offers many useful built-in features not available in 16-bit WinHelp, including full-text indexing and search capability.

Winsock An abbreviation for Windows Sockets, a networking API for implementing Windows applications that use TCP/IP, such as FTP and Telnet.

workstation A client computer on a LAN or WAN that is used to run applications and is connected to a server from which it obtains data shared with other computers. It is possible, but not common, for some network servers to be used as both a server and a workstation. Microsoft Windows NT, for instance, permits this. Workstation is also used to describe a high-priced PC that uses a proprietary microprocessor and proprietary architecture to create what some call an *open system*.

Oracle Ex. 1009, p. 1212
Oracle v. DeBellis
IPR2024-00221

WOSA Acronym for the Windows Open Services Architecture that is the foundation for such APIs as ODBC, MAPI, and TAPI. Microsoft also develops special vertical-market WOSA APIs for the banking, financial, and other industries.

WOW An acronym for Windows on Win32, a subsystem of Windows NT that allows 16-bit Windows applications to run in protected memory spaces called *virtual DOS machines* (VDMs).

xBase Any language interpreter or compiler or a database manager built upon the dBASE III+ model and incorporating all dBASE III+ commands and functions. Microsoft's FoxPro and Computer Associates' Clipper are xBase dialects.

Yes/No field A term used by Access to describe a field of a table whose allowable values are Yes (True) or No (False). Yes/No fields are called *logical* or *Boolean fields* by Access 95.

Oracle Ex. 1009, p. 1213
Oracle v. DeBellis
IPR2024-00221

Appendix B

Naming Conventions for Access Objects and Variables

by Stan Leszynski, Leszynski Company, Inc., and Kwery Corporation

This appendix describes the Leszynski Naming Conventions (LNC), a set of standardized approaches to naming objects during Access development. These naming conventions were born of necessity, because some members of the staff of my firm spend all day in Access development, year after year. They were also born of a different need—a void that existed in the marketplace due to a lack of consensus about development styles among leading Access developers.

I am grateful to Greg Reddick, a former contractor with one of my companies, for jointly authoring the Access 1.x and 2.0 versions of our Access naming conventions with me (referred to as *L/R* for *Leszynski/Reddick*). The L/R conventions were distributed broadly, with more than 500,000 copies in print, and have become the most widely used such conventions in the Access community. During the last few years, we have received feedback about L/R from hundreds of developers and companies, and have tried to accommodate some of their input—as well as our ongoing experiences—into LNC.

LNC improves upon the previous Access style by considering developers who work with multiple Microsoft development tools. Access, Visual Basic, Excel, and other Microsoft products such as Visual Basic Script (VBScript) have more in common in their Windows 95 and Windows NT 4.0 versions than in any previous iterations. Consequently, this Access style dovetails with the LNC development style for all of VBA and FoxPro.

The prefixes, tags, and qualifiers in the naming conventions described in this appendix that are derived from L/R are underlined. If you are a user of the Access 2.0 version of L/R, new additions for you to note are those conventions that are not underlined. I use the terms *naming conventions*, *style*, and LNC interchangeably throughout this chapter.

Oracle Ex. 1009, p. 1214
Oracle v. DeBellis
IPR2024-00221

Note

Publishing limitations preclude reprinting of the complete text of LNC in this appendix. To obtain the unabridged version of the LNCs, see the section "How to Get the Complete Text of the LNCs" at the end of this appendix.

The Leszynski Naming Conventions for Microsoft Access: A Primer

Naming conventions are one of the foundation elements of your overall development style. The naming conventions were developed primarily to achieve four objectives:

- To enable you to quickly understand an application's structure and code by making object names more informative.
- To simplify team development of applications by creating a standardized vocabulary for all team members.
- To improve your ability to work with Access objects, including enforcing object name sort orders, creating self-documenting program code, and enhancing find and replace capabilities.
- To increase your ability to create tools for our Access development work, and to create code libraries across various VBA platforms.

To meet these objectives, you create and apply consistent naming conventions to the following Access objects:

Tables	Reports	Variables
Table fields	Report controls	Constants
Queries	Macros	User-defined types
Forms	Modules	
Form controls	Procedures	

Object names are the foundation upon which your entire application is built, so they are almost impossible to change once development has begun in earnest. Therefore, you will probably not find it cost- or time-efficient to retrofit these conventions into your existing applications. For new applications, however, you should apply these naming conventions consistently from the moment you create your first object in a new Access database file.

Your naming conventions rely primarily on *leading tags*—several characters placed before an object's name (for example, `qryOrderByMonth`). This approach is sometimes referred to as *Hungarian Notation*. Leading tags provide several benefits:

Oracle Ex. 1009, p. 1215
Oracle v. DeBellis
IPR2024-00221

- The first item you observe about an object when you see its name is the leading type tag, which is often more important than the name itself.
- Leading tags drive the ordering of object names in Access lists, sorting by type and then by base name.
- Leading tags are consistently located in the same place in an object's name, making them easier to find by parsers and other tools.

Note

The term *Hungarian* refers to the nationality of Charles Simonyi, a programmer at Microsoft who wrote a doctoral thesis titled "Program Identifier Naming Conventions," in the 1980s.

If you are averse to Hungarian Notation for some reason and prefer trailing tags, LNC can still work for you. However, LNC prescribes no standard for locating and punctuating trailing tags. You have to decide if they are offset with an underscore (`OrderByMonth_qry`), capitalization (`OrderByMonthQry`), or some other technique.

Using trailing tags on database objects is problematic for you when your application also contains Visual Basic for Applications (VBA) code. The primary justification given by developers who prefer trailing tags on database objects is that it allows the objects to sort by base names rather than tags in ordered lists. However, such developers often still use leading tags and prefixes in their VBA code, because there seems to be no compelling argument that can be made for trailing tags on VBA objects such as variables. If you mix your styles like this, be prepared to justify your lack of consistency.

Because some developers, especially newer ones, prefer to minimize the complexity of a naming convention, LNC provides the following two levels for Access users:

- Level One has the minimum realistic subset of tags, but consequently provides lesser detail about the application. It is intended for users whose work is centered around the Database window and who do not develop applications but only database objects.
- Level Two provides greater detail and the flexibility to create your own extensions. It is intended for application developers.

Access Object Types

For purposes of this appendix, the standardized terminology for grouping objects was created. These group names are used when discussing naming conventions:

- Database objects:

Tables	Queries forms	Macros
Table fields	Reports	Modules

Oracle Ex. 1009, p. 1216
Oracle v. DeBellis
IPR2024-00221

- Control objects:

  Form controls Report controls

- VBA objects:

  Procedures Constants

  Variables User-defined types

Structuring Object Names

In *LNC*, object names are constructed using this syntax for Level One:

[*prefix(es)*] [*tag*] *BaseName* [*Qualifier*] [*Suffix*]

For Level Two, the syntax varies slightly:

[*prefix(es)*] *tag* [*BaseName*] [*Qualifier*] [*Suffix*]

The brackets indicate optional syntax elements. Notice that, for Level One, the `BaseName` element is required and the `tag` is optional in some cases. At Level Two, the `tag` element is required even though the `BaseName` is not in some cases. These options are explained in the sections "What is a Tag?" and "Creating Database Object Base Names" later in this appendix.

Note

In the syntax diagrams, the case of each element reflects its case in actual use. The element `tag` is lowercased because the tags themselves are always found in lowercase.

Table B.1 shows sample object names using these constructions.

Table B.1 Object Names Constructed in LNC Format

Object Name	Prefix	Tag	Base Name	Qualifier	Suffix
`tblCust`		`tbl`	`Cust`		
`qsumSalesPerfBest_WA`		`qsum`	`SalesPerf`	`Best`	`_WA`
`plngRecNumMax`	`p`	`lng`	`RecNum`	`Max`	
`ialngPartNum`	`ia`	`lng`	`PartNum`		

What Is a Prefix?

A *prefix* is an identifier that precedes a tag and clarifies it narrowly. Prefixes describe one or more important properties of an object. For example, a `Long` variable that is public in scope (declared `Public`) has a prefix `p`, as in `plngRecNumMax`. Prefixes are one or two characters long and lowercased. Multiple prefixes can be used together on one object.

Oracle Ex. 1009, p. 1217
Oracle v. DeBellis
IPR2024-00221

What Is a Tag?

A *tag* is a multi-character phrase placed against an object base name to characterize it. In object-oriented programming terms, the tag is basically an identifier for the class. At Level One, you could say that tags define an object's general class; for example, `qry` for a query of any type. At Level Two, the tag defines the specific class; for example, `qdel` defines a delete query.

> **Note**
>
> The word *class* here refers to a naming convention construction, not an exact object model construction. For example, there is only one `Query` (or `QueryDef`) class object in Access, and the data action (delete, update, and so on) is determined by its SQL statement, not its class. LNC prescribes several tags for this one Access class.

Tags are three or four characters long for readability purposes and to allow for the hundreds of combinations necessary as the Office object model grows over time. They are always to the left of the base name and lowercased, so that your eye reads past them to the beginning of the base name.

A tag is created to mnemonically represent the word it abbreviates, such as `frm` for form. However, some tags may not seem fully mnemonic for two reasons:

- The perfect (or obvious) tag for a particular object may already be assigned to another object.
- Where common objects (objects with similar properties and usage) exist in multiple Microsoft applications, the tag for one may be used to represent similar objects in other products, even if the names are different. For example, an Access `Rectangle` object is almost identical in structure and purpose to a Visual Basic `Shape` object. Because the Visual Basic conventions have existed longer than your Access conventions, you use the Visual Basic `Shape` object tag you were already using, `—shp—`to represent Access rectangles.

What Is a Base Name?

The *base name* is the starting point when you name a particular object—the name you would use anyway if you had no naming conventions. The LNC guidelines for creating base names are driven by a set of rules stated in the following sections.

What Is a Qualifier?

A *qualifier* is an extension following the base name that provides context to the specific use of an object. Unlike prefixes, which detail properties of the object (for example, that the variable has public scope), qualifiers describe how the object is being used in a context. For example, `plngRecNumMax` is obviously the maximum record number in an

Oracle Ex. 1009, p. 1218
Oracle v. DeBellis
IPR2024-00221

application that could also have variables for the minimum (`plngRecNumMin`) and current (`plngRecNumCur`) record numbers. Qualifiers are short and written with mixed upper- and lowercase (see Table B.2).

Table B.2 Standard LNC Qualifiers

Qualifier	Usage	Qualifier	Usage
`Cur`	Current element of a set	`Next`	Next element of a set
`Dest`	Destination	`New`	New instance or value
`First`	First element of a set	`Old`	Prior instance or value
`Hold`	Hold a value for later	`Prev`	Previous element re-use of a set
`Last`	Last element of a set	`Src`	Source
`Max`	Maximum item in a set	`Temp`	Temporary value
`Min`	Minimum item in a set		

What Is a Suffix?

Suffix elements provide specific information about the object and are only used as "tie-breakers" when more detail is required to differentiate one object name from another. These are the only elements in the syntax where your naming conventions do not specify standardized values. You will create suffix items as needed by your company, development team, or application. For example, a series of queries that summarized the best sales performance by state would need the state name in each object name to properly qualify it, as in `qsumSalesPerfBest_AK`. Placing the state name at the very end of the name as a suffix item allows the entire collection of related queries to sort together, like this:

```
qsumSalesPerfBest_AK
qsumSalesPerfBest_AL
...
qsumSalesPerfBest_WY
```

Because the suffix is the last piece of information on a name, it can be easier for you to find if delimited from the rest of the object name with an underscore, as shown here. Use of the underscore is optional, not required.

Creating Database Object Base Names

The building blocks of your Access application are its database objects. When creating base names for these objects, you should give careful consideration to the purpose of the object, the approaches used to name associated objects, and the rules of thumb that follow for naming database objects.

Rules for Base Names

Follow these rules when developing a base name for a new database object:

- Spaces are not allowed in any object name. Spaces create a multitude of problems with consistency, readability, and documentation. Where the readability of a space is required, use an underscore instead.

Oracle Ex. 1009, p. 1219
Oracle v. DeBellis
IPR2024-00221

- Object names begin with a letter and should only include letters, digits, and underscores. The use of special characters in object names is disallowed to comply with the naming rules of both VBA and Microsoft SQL Server. This allows your Basic variable names to include database object base names, and your entire Access schema to be easily upsized to the more powerful Microsoft SQL Server platform.
- Object names use mixed upper- and lowercase to add readability to the name. (Previously, some developers used all lowercased names to allow for upsizing to Microsoft SQL Server. Starting with SQL Server Version 6.0, that product is now installed case-insensitive and allows you to maintain upper and lower case in object names that are moved to the server from Access.)
- The only syntax element that can have multiple capital letters is the base name. A qualifier or suffix begins with a single capital letter and then contains only lowercased letters, unless it is an abbreviation, as in `qsumSalesPerfBestUSA`. If you need to clearly see the elements of a name (prefixes, tag, base name, qualifier, and suffix), LNC permits—but does not require—underscores as separators, as in `qsum_SalesPerf_Best_USA`.
- Object names are usually singular (`Widget`) rather than plural (`Widgets`). By implication, tables, queries, forms, and reports are plural; they usually work with more than one record, so why restate the obvious?
- An object's base name should include the base names of any table objects it is built on, if practical. This rule is explained later in this section.

The first and second bullets also apply to the other naming convention elements: prefixes, tags, qualifiers, and suffixes. These elements should never include spaces or special characters.

You should abbreviate object base name elements wherever possible using a standardized abbreviation table. You can extend LNC with your own standard abbreviations as well. You should create and use standardized terminology in your applications wherever possible.

Base Name Length Limits

LNC includes some constraints and suggestions for object name lengths. You should target table name lengths at a 15-character maximum, for two reasons:

- Short names (15 characters or less) fully display within the default column width of the Access query design grid.
- Query, form, and report names usually include the base name(s) of the primary table object(s) they relate to, and will be unusably long if the table base names are long.

Beyond the 15-character target, you should absolutely limit table name lengths to 30 characters, which maintains compatibility with the table name length limit in SQL Server. For other objects, you should target a 30-character limit as well, because Access shows no more than the first 30 characters of object names in the default width of any of its lists or property grids.

Oracle Ex. 1009, p. 1220
Oracle v. DeBellis
IPR2024-00221

Compound Base Names

The name of an object that is driven by a table must include the base name of the table. Thus, for the `tblCust` table, the primary query would be `qryCust`, the primary form `frmCust`, and so forth. Queries, forms, and reports that are sourced from multiple tables should reflect the base names of all the tables if it is practical. If not, you must decide which tables are the primary tables and list as many as possible in the name.

Generally, in a multitable query, form, or report, the most important tables are not necessarily the first and second, but more often the first and last. So, a query joining `tblCust` to `tblAddr` to `tblPhone` to get the phone numbers for customers would be named `qryCustAddrPhone` if the address information is included in the query result, or simply `qryCustPhone` if the address information is used to join to the phone numbers and is not displayed.

Bound control base names on forms and reports are always equivalent to the base name of the bound field (the ControlSource). For example, a text box tied to the LastName field is named `txtLastName`.

Field Base Names

As a part of standardizing terminology, you should adhere to the concept of a centralized data dictionary. This principle dictates that any fields in the data structure that have the same name must have the same properties and data purpose. For example, if the LastName field in `tblCust` is of type Text 30, and holds the customer last name, any other field named LastName in the same application must have the same type, length, properties, and purpose. If your application needs last name fields for both customers and dealers, this philosophy dictates that you name them differently (such as CustLastName and DlrLastName).

Applying the centralized data dictionary principle also means that table fields do not get leading prefixes or tags, because your data dictionaries should be platform-neutral. That way, a field does not have to be renamed if data is upsized or ported to a platform with different data types. A table is still called a table in SQL Server, so moving `tblCust` there from Access would require no rename. However, if `tblCust` had a field `lngCustID` defined as a Long Integer in Access, moving the database to SQL Server would require a field rename to `intCustID`, because SQL Server uses the data type name Integer to mean the same as Access's Long Integer. Renaming fields affects all dependent objects and code, so should be avoided at all costs. You would call the field simply `CustID`.

Qualifiers and suffixes are acceptable in field names, however, because they describe the object's data purpose and not its type.

Ordering Base Name Elements

Object base name elements should be ordered from left to right with respect to their importance, readability, and desired sort order. In the example from the previous paragraph, CustLastName is a better name than LastNameCust, because the group name portion (`Cust` or `Dlr`) carries greater weight in an object's name than the specific item

Oracle Ex. 1009, p. 1221
Oracle v. DeBellis
IPR2024-00221

name (LastName or PhoneNum). Think of Cust as the name of a collection of customer-related items and this rule becomes clear—what you are really saying is that CustLastName is analogous to `Cust(LastName)` or `Cust.LastName` in `Collection.Object` terminology.

You can naturally carry this example to its extreme and say that the `Customers` collection really has a `Names` collection with multiple elements, including Last, thus the representation of that idea as `Cust.Name(Last)` would lead to the field name CustNameLast instead. Such a construction model still fits within the rules of LNC, and there's no reason not to use it. In practice, however, such names often become fairly unreadable, even if they are accurate.

Naming Conventions for Database Objects

In Level Two of LNC, tags are required for all of the Access (and Jet) database objects listed in Tables B.1 and B.2. Level One also recommends that you place tags on every object name, but recognizes that non-developers may prefer to save time, effort, and complexity by leaving tags off of objects where the context is obvious while viewing the Database window. Thus, Level One users are required only to place the `qry` tag on queries in order to differentiate them from tables in any combined lists, such as the "Choose the table or query..." combo box on Form and Report Wizards. Placing tags on the other objects listed is recommended, but optional.

Caution

It can be difficult to propagate name changes throughout a database, so if you are a casual user now but expect to become a developer later—and thus migrate from Level One of LNC to Level Two—you would be unwise to leave tags off of any object names. Use the Level One tags now on all objects.

Tags for Database Window Objects

Table B.3 lists the Level One tags for Database window objects. Note that only one tag exists for each object type.

Table B.3 Level One Database Window Object Tags

Object	Tag	Object	Tag
Class	`cls`		
Form	`frm`	Report	`rpt`
Macro	`mcr`	Subform	`fsub`
Module	`bas` or `mdl`	Subreport	`rsub`
Query	`qry`	Table	`tbl`

Though Level One is the simplified naming model, you provide tags to identify subform and subreport objects specifically. The distinction between objects and subobjects is

Oracle Ex. 1009, p. 1222
Oracle v. DeBellis
IPR2024-00221

critical for non-developers who navigate using the Database window. Because it is not appropriate to open subforms and subreports directly from the Database window, they must be clearly identified and grouped using tags.

Table B.4 lists the Level Two tags for Database window objects.

Table B.4 Level Two Database Window Object Tags

Object	Tag	Object	Tag
Class module	cls		
Form	frm	Query (select)	qsel
Form (class module)[1]	fcls	Query (SQL pass-through)	qspt
Form (dialog)	fdlg	Query (union)	quni
Form (lookup table)[2]	flkp	Query (update)	qupd
Form (menu/switchboard)	fmnu	Report	rpt
Form (message/alert)	fmsg	Report (detail)	rdet
Form (subform)[3]	fsub	Report (sub)[3]	rsub
Form (wizard main)	fwzm		
Form (wizard subform)	fwzs		
Macro	mcr	Report (summary)	rsum
Macro (for form/report)	m[*obj*]	Table	tbl
Macro (bar menu)	mmbr	Table (attached Btrieve)	tbtv
Macro (general menu)	mmnu	Table (attached dBASE)	tdbf
Macro (shortcut menu)	mmct	Table (attached Excel)	txls
Macro (submenu/dropdown)	mmsb	Table (attached Fox)	tfox
Module[4]	bas	Table (attached Lotus)	twks
Query	qry	Table (attached ODBC)	todb
Query (form/report source)	q[obj]	Table (attached Paradox)	tpdx
Query (append)	qapp	Table (attached SQL Server)	tsql
Query (crosstab)	qxtb	Table (attached text)	ttxt
Query (data definition)	qddl	Table (audit log)	tlog
Query (delete)	qdel	Table (lookup)[2]	tlkp
Query (form filter)	qflt	Table (many-to-many relation)	trel
Query (lookup table)[2]	qlkp	Table (summary information)	tsum
Query (make table)	quak		

1. Access does not implement class modules formally as an object type, as Visual Basic does.
2. A lookup table has records that map short codes to full text values, like state abbreviations to states, and is used to populate combo and list boxes, validate fields, and so forth.
*3. A single subform or subreport may be used in several different parent objects, thus naming subobjects is not as simple as just adding **_Sub** to the end of the parent's base name (or some similar technique).*
4. You use the module tag bas to maintain consistency with the file name extension used by Visual Basic modules.

The tags for Level Two provide rich detail about the objects and sort objects with similar attributes together. For example, lookup tables and their maintenance forms are often

Oracle Ex. 1009, p. 1223
Oracle v. DeBellis
IPR2024-00221

used over and over in multiple applications. The tags `tlkp`, `qlkp`, and `flkp` clearly identify these objects, making it easy for you to import them from an existing database into a new one when using the object list in Access's Import dialog. However, if a particular database does not warrant rich detail, you have generic tags to use as well (for example, `qry` instead of `qsel`).

In two special cases, the conventions prescribe a single character tag added to the front of the full object name (including the tag) of the related object. This situation occurs where a macro is created solely for a particular form or report, as in `mfrmCust`, and a query is created solely to serve as the `RecordSource` property for one particular form or report, as in `qfrmCust`.

Tags for Form and Report Control Objects

Table B.5 lists the Level One tags for control objects on forms and reports.

Table B.5 Level One Form and Report Control Object Tags

Control	Tag
Label	`lbl`
Other types	`ctl`

These Level One control tags provide no differentiation of control type other than to distinguish labels (which do not interact with the user) from controls that can display or modify data. This level of detail is not adequate for applications where VBA codewill be written behind forms or reports, but can be a convenience with macro-centric applications.

Table B.6 lists the Level Two tags for control objects on forms and reports. A different tag is provided for each built-in control type. Tags also can be created for standard OLE controls. VBA code written behind forms and reports using this convention will reflect a control's type in its event procedure names (for example, `cboState_AfterUpdate`). The automatic sorting provided by this notation in the Access module design window can be very helpful during development. The value returned by the `TypeOf()` function for an Access control is a reflection of its class.

Tip

Make sure all control tags are three characters long so objects sort correctly.

Table B.6 Level Two Form and Report Control Object Tags

Control	Tag	TypeOf
Bound object frame	`frb`	`BoundObjectFrame`
Chart (graph)	`cht`	`ObjectFrame`
Check Box	`chk`	`CheckBox`
Combo box	`cbo`	`ComboBox`

(continues)

Oracle Ex. 1009, p. 1224
Oracle v. DeBellis
IPR2024-00221

Table B.6 Continued

Control	Tag	TypeOf
Command button	cmd	CommandButton
Custom control	ocx	CustomControl
Hyperlink	hlk	Hyperlink
Image	img	Image
Label	lbl	Label
Line	lin	Line
List box	lst	ListBox
Option button	opt	OptionButton
Option group	grp	OptionGroup
Page break	brk	PageBreak
Rectangle	shp	Rectangle
Section	sec	Section
Subform/Subreport	sub	Subform
Tab	tab	TabControl
Text box	txt	TextBox
Toggle button	tgl	ToggleButton
Unbound object frame	fru	ObjectFrame

Using Menu Macros



Access 97 replaces menu macros used by previous versions of Access with command bars. Thus, this section applies only to existing Access applications that you upgrade to Access 97 and retain their menu macros.

Menu macros behave differently than standard macros; thus, they fall under separate guidelines when creating their names. Menu macros are either used for bar menus or for shortcut menus.

If you use the Menu Builder Add-In, your custom menu macros are assigned names when the builder saves your menu design. You have little control over anything but the tag and base name for the primary (bar) menu. If you use the LNC convention of `mmnuname` for your menu macro, the builder creates an entire tree of macros beginning with the string you enter, for example:

```
mmnuMain
mmnuMain_File
mmnuMain_File_Print
```

If you prefer to create and name menus yourself, you can use a similar convention, but with more explicit tags. Bar menus should be prefixed with `mmbr`, sub (drop-down) menus with `mmsb`, and shortcut menus with `mmct`. Detailed tags like this help greatly when you are selecting from a list of menus to assign to the `MenuBar` or `ShortcutMenuBar` properties of a form. This convention also sorts menus by type in the Database window. The macro listing shown here, if produced manually rather than with the builder, would look like this:

Oracle Ex. 1009, p. 1225
Oracle v. DeBellis
IPR2024-00221

```
mmbrMain
mmctMain_File
mmsbMain_File
mmsbMain_File_Print
```

Any saved submenu macro can be selected as the primary object in a shortcut menu, so this convention gives you the flexibility shown here to create component submenus, which can be called from both multiple bar menus and multiple shortcut menus, and thus reused.

Creating VBA Object Base Names

When creating VBA object base names, remember that the base name must be descriptive even in the absence of its tag. For some programmers, the syntax `Dim I As Integer` for a loop variable is quite acceptable. Within LNC, however, the variable `I` would become `iintLoop`. Single-character variable names, especially without tags, are not allowed. Instead, create a list of short and standardized work variables to handle common daily needs. The following sections describe the LNC rules for creating VBA object base names.

Rules for Base Names

Creating VBA object base names involves following the same rules listed earlier for creating database object base names:

- Spaces are not allowed in any object name.
- Object names begin with a letter and should only include letters, digits, and underscores.
- Object names use mixed upper-and lowercase to add readability to the name.
- The only non-abbreviated syntax element that can have multiple capital letters is the base name.
- Object names are usually singular rather than plural.
- An object's base name should include the base names of any objects it is built on, if practical.

Note

The final rule is an expanded version of the corresponding rule for database objects, which stated that table base names should propagate into names of dependent objects. In Visual Basic, that rule expands to require a reference in variable names to objects of any type that they relate to. For example, a Recordset variable created on `tblCust` should be named `rstCust`. Also, if a string array variable of part numbers `astrPartNum` had an Integer index variable, it should include the array's base name in its own: `iaintPartNum`.

Base Name Lengths

There is no LNC rule limiting variable name length, but common sense dictates that variable names longer than 15 or 20 characters waste a lot of keystrokes at each use.

Oracle Ex. 1009, p. 1226
Oracle v. DeBellis
IPR2024-00221

For procedure names, the VBA Module Editor by default shows the first 30 characters of a procedure name, so this number is suggested as the target maximum procedure name length.

Tip

Abbreviate VBA object base name elements wherever possible using a standardized abbreviation table. You can extend LNC by creating your own standard abbreviations as well. You should create and use standardized terminology in your applications wherever possible.

Compound Base Names

Procedure base names should follow the construction `ObjectVerb`, where the `Object` portion describes the primary object type affected (often the same as the primary argument), and `Verb` describes the action. This style sorts functions and subs by their target object when shown in ordered lists:

```
FormCtlHide
FormCtlShow
FormPropAdd
FormPropGet
FormPropSet
```

This sort order is much more appealing than the more common alternative with `VerbObject` construction:

```
AddFormProp
GetFormProp
HideFormCtl
SetFormProp
ShowFormCtl
```

Naming Conventions for VBA Objects

In Level Two of LNC, tags are required for the following VBA objects:

- Variables
- Type structures
- Constants

Optional tags are also available for some types of procedures. By definition, if you are a Level One user, LNC assumes that you are not writing VBA code. If you are creating procedures, you are a Level Two user and should always apply Level Two tags and prefixes to database objects as well as VBA objects.

In the syntax diagram earlier, you saw that base names are optional in some Level Two constructions. When programming in VBA in Level Two, the `tag` element is always required, but the `base name` is optional for local variables only. For example, a procedure that declared only one form object variable could legitimately use the variable name `frm`, which is a tag without a base name. Type structures, constants, and variables that have module-level or public scope must have both a tag and base name.

Oracle Ex. 1009, p. 1227
Oracle v. DeBellis
IPR2024-00221

Tags for Variables

Visual Basic variable tags are noted in Tables B.7–B.9, grouped by type of variable.

Table B.7 Tags for VBA Data Variables

Variable Type	Tag	Variable Type	Tag
Boolean	bln	Integer	int
Byte	byt	Long	lng
Conditional Compilation Constant	ccc	Object	obj
Currency	cur	Single	sng
Date	dtm	String	str
Double	dbl	User-Defined Type	typ
Error	err	Variant	var

Note

Many developers use dat as a tag for date objects, but this conflicts for us with the tag for Visual Basic's Data control.

In the preceding table, note that Conditional Compilation Constant, Error, and User-Defined Type are not true data types (created with `Dim Name As Datatype`), but rather programming concepts. A Conditional Compilation Constant variable is a flag of type `Boolean`, an Error variable is a `Variant` created with the `CVErr()` function, and user-defined types are unique constructs. Table B.8 lists examples of tabs for VBA object variables.

Table B.8 Tags for VBA Object Variables

Object	Tag	Object	Tag
Access.Application	acca	Graph.Application	gpha
Assistant	ast		
Application	app	Graph.Chart	gphc
Collection	col	GroupLevel	lvl
Control	ctl	MSProject.Application	prja
Controls	ctls	MSProject.Project	prjp
CommandBar	cbr	Module	bas or mdl
CommandBars	cbrs	Modules	bass or mdls
		Page	pge
		Pages	pges
CustomControl	ocx	PowerPoint.Application	ppta
		Reference	ref
		References	refs

(continues)

Oracle Ex. 1009, p. 1228
Oracle v. DeBellis
IPR2024-00221

Table B.8 Continued

Object	Tag	Object	Tag
CustomControlInReport	ocx	Report	rpt
Excel.Application	xlsa	Reports	rpts
Excel.Chart	xlsc	SchedulePlus.Application	scda
Excel.Sheet	xlsw	Word.Application	wrda
Form	frm	Word.Basic	wrdb
Forms	frms		

Table B.8 mixes entry points for OLE automation server objects, such as Excel worksheets and Project projects, with objects internal to Access, such as collections. While all of these items are objects, the two object types are treated differently in code.

Variables for objects in the Access object hierarchy can be dimensioned directly by class, as in this line:

```
Dim colFrmBldr As Collection
```

OLE Automation variables are created and used with a different syntax, as in this example:

```
Dim oxlsa As Object
Set oxlsa = CreateObject("Excel.Application")
```

The naming convention for entry points into OLE Automation server applications follows this syntax:

```
applicationtag entrypoint
```

where *applicationtag* is three characters and *entrypoint* is a single character. See the section "Creating Your Own Tags" later in this appendix for a complete explanation of this syntax. Note also that OLE Automation object variables are prefixed with o, as described in the "Prefixes for Variables" section that follows.

Table B.9 lists the LNC tags for Access DAO (data access object) variables.

Table B.9 Tags for Data Access Object Variables

Object	Tag	Object	Tag
Connection	cnc	QueryDef (Paradox)	qpdx
Connections	cncs	QueryDef (SQL Server)	qsql
Container	con	QueryDef (Text)	qtxt
Containers	cons	QueryDefs	qdfs
DBEngine	dbe	Recordset (Any type)	rst
Database (Any type)	dbs	Recordset (Btrieve)	rbtv
Database (Btrieve)	dbtv	Database (Excel)	dxls
Database (dBASE)	ddbf	Recordset (dBASE)	rdbf

Oracle Ex. 1009, p. 1229
Oracle v. DeBellis
IPR2024-00221

Object	Tag	Object	Tag
Database (FoxPro)	dfox	Recordset (Dynaset)	rdyn
Database (Jet)	djet	Recordset (Excel)	rxls
Database (Lotus)	dwks	Recordset (FoxPro)	rfox
Database (ODBC)	dodb	Recordset (Lotus)	rwks
Database (Paradox)	dpdx	Recordset (ODBC)	rodb
Database (SQL Server)	dsql	Recordset (Paradox)	rpdx
Database (Text)	dtxt	Recordset (Snapshot)	rsnp
Databases	dbss	Recordset (SQL Server)	rsql
Document	doc	Recordset (Table)	rtbl
Documents	docs	Recordset (Text)	rtxt
Dynaset	dyn	Recordsets	rsts
Error	err	Relation	rel
Errors	errs	Relations	rels
Field	fld	Snapshot	snp
Fields	flds	Table	tbl
Group	gru	TableDef (Any type)	tdf
Groups	grus	TableDef (Btrieve)	tbtv
Index	idx	TableDef (dBASE)	tdbf
Indexes	idxs	TableDef (Excel)	txls
Parameter	prm	TableDef (FoxPro)	tfox
Parameters	prms	TableDef (Jet)	tjet
Property	prp	TableDef (Lotus)	twks
Properties	prps	TableDef (ODBC)	todb
QueryDef (any type)	qdf	TableDef (Paradox)	tpdx
QueryDef (Btrieve)	qbtv	TableDef (SQL Server)	tsql
QueryDef (dBASE)	qdbf	TableDef (Text)	ttxt
QueryDef (Excel)	qxls	TableDefs	tdfs
QueryDef (FoxPro)	qfox	User	usr
QueryDef (Jet)	qjet	Users	usrs
QueryDef (Lotus)	qwks	Workspace	wsp
QueryDef (ODBC)	qodb	Workspaces	wsps

In Tables B.8 and B.9, tags for collection variables are made by adding s after the tag for the object type stored in the collection. The tags dyn, snp, and tbl above for Dynaset, Snapshot, and Table objects are directly relevant to users of Access 1.x and Access 2.0. Starting with Access 95, these object types are allowed only as a subtype of recordset variables, thus the recordset tags rdyn, rsnp, and rtbl.

Oracle Ex. 1009, p. 1230
Oracle v. DeBellis
IPR2024-00221

Note

Although you saw earlier that a tag by itself is a legitimate variable name, a few variable tags shown (such as `int`) are VBA reserved words and will not compile in your procedures. Such tags require a base name.

Prefixes for Variables

The prefixes for Visual Basic variables can be categorized into two groups: prefixes for scope, and all other prefixes. Because the model for variable scope has changed somewhat in Access 95, you examine scope prefixes first. The prefixes in the following list are ordered by increasing (broader) scope:

- *No Prefix*. Use no prefix for variables that are local to a procedure.
- `s`. Place this prefix before variables that are declared locally to a procedure with a Static statement.
- `m`. Use this prefix for module-level variables that are declared with **`Dim`** or **`Private`** statements in the Declarations section of a module.
- `p`. Use this prefix to denote variables declared as **`Public`** in the Declarations section of a form or report module. Such variables are publicly available to other procedures in the same database only.
- `g`. Use this prefix to denote variables declared as **`Public`** in the Declarations section of a standard module. Such variables are truly global and may be referenced from procedures in the current or other databases.

When used, scope prefixes always begin a variable name and precede any other prefixes.

In addition to scope, there are other characteristics of variables that can be identified by prefixes, as in the following list:

- `a`. Use this prefix to denote a variable that is declared as an array, including a `ParamArray` argument to a function.
- `c`. This prefix is placed before constants defined by the user.
- `e`. Use this prefix for a variable that is an element of a collection. Such variables are usually part of a **`For Each...Next`** loop structure.
- `i`. Use this prefix to denote a variable (usually of type **`Integer`**) that serves as an index into an array or an index counter in a **`For...Next`** loop.
- `o`. This prefix is placed before object variables that reference OLE Automation servers, where the tag denotes the type of server.
- `r`. Use this prefix for variables that are arguments (parameters) passed in to a procedure and declared **`ByRef`**, or not declared as either **`ByRef`** or **`ByVal`** (including a `ParamArray`), which implies **`ByRef`**.

Oracle Ex. 1009, p. 1231
Oracle v. DeBellis
IPR2024-00221

- t. Use this prefix to describe a variable that is declared as a user-defined **Type** structure.
- v. Use this prefix for variables that are arguments (parameters) passed in to a procedure and declared **ByVal**.

A prefix provides a very detailed description of a variable, so the number of allowable prefix combinations is limited, as shown in Table B.10.

Table B.10 Allowable Prefix Combinations

Any One of These...	...Can Come Before This
s, m, p, g, r, v	a
m, p, g	c
s, m, p, g, r, v	e
s, m, p, g, r, v	i
s, m, p, g, r, v	ia
s, m, p, g, r, v	o
m, p, g	t

Variables require a unique prefix when declared Public in a widely distributed application to prevent name contentions. See the section "Tags and Prefixes for Procedures" later in this chapter for more information.

Naming Constants

Access 95 introduced sweeping changes in the area of constants. The changes most relevant to naming conventions include these:

- A constant can now be assigned a data type when it is defined.
- All Access constants have been renamed and carry a tag of ac, db, or vb to identify their primary functional area, Access, Jet, or VBA, respectively.
- Constants can now be created with the **Variant** data type.

When creating constants, use a scope prefix (if appropriate), the prefix c, and the suitable tag for the constant's data type. To properly synchronize the tag and the data type, do not let Access assign the type; always use the full **Const** *Name* **As** ***Datatype*** syntax.

Constants require a unique prefix when declared **Public** in a widely distributed application to prevent name contention. See the following sections for more information on name contention.

Tags and Prefixes for Procedures

Whether and how to prefix and tag procedure names is a debatable subject. In general, this style neither requires nor encourages placing characters before a procedure name except in the following situations.

Oracle Ex. 1009, p. 1232
Oracle v. DeBellis
IPR2024-00221

Prefixes for Procedures Procedures can have scope similar to that of variables:

- s (**Static**)
- m (**Private**)
- p (**Public**)
- g (global **Public**)

LNC supports the use of these scope prefixes on function names if they solve a particular need and are used consistently throughout an application.

If you are creating software for retail sale, for inclusion in the public domain, or that will be broadly distributed in some other manner, LNC requires that you prefix **Public** variables, constants, and procedures with a unique author prefix identifying you or your company. The prefix consists of two or three unique characters and an underscore, and prevents your object names from conflicting with object names in other referenced or referencing databases on a user's machine.

To create an author prefix, use your personal or company initials. For example, author prefixes for my companies are `lci` for Leszynski Company, Inc., and `kwc` for Kwery Corporation. Before using your selected prefix, make an effort to determine if the prefix is already widely in use. Kwery Corporation maintains a registry of VBA prefixes for Office add-in vendors.

Tags for Procedures The LNC style prescribes the following naming convention tags for procedures:

- `cbf`. Use this tag on procedure names for code behind a form or report. This tag clearly differentiates such procedures from `Property` procedures and event procedures.
- `prp`. Use this tag on `Property` procedure names defined with `Property Get`, `Property Let`, and `Property Set` statements. This tag clearly differentiates such procedures from functions, subprocedures, and event procedures.

LNC does not require or suggest assigning a data type tag to functions to reflect their return value. However, if you have a specific need to tag procedures to reflect their return value type, use the appropriate tags from the preceding section "Tags for Variables" and apply the tags consistently to all procedures in an application.

Using Macros Instead of VBA

If you use Access macros, apply the rules previously described for VBA procedure base names to your macro base names. Macro groups in the Database window should also utilize the macro prefixes and tags noted earlier in this appendix. Individual macros within a macro group do not have prefixes or tags, except for event macros as noted later on.

All macros for a specific form or report should be placed into one macro group, with the form or report name as the macro base name. Within the macro group, create standard macros and event macros for the form or report. If a macro is tied to a form or report event, show the event name or an abbreviation in the macro name. For example, you would store macros related to `frmCust` in macro group `mfrmCust`. Event macro names in

Oracle Ex. 1009, p. 1233
Oracle v. DeBellis
IPR2024-00221

the group might include `Form_Current` and `txtLastName_Change`.

Macros that are not specific to a form or report should hold actions that have some common functionality or purpose. Use the same grouping methodology (related items together) that you would apply to locate procedures in standard VBA modules.

VBA Object Name Examples

Table B.11 shows examples of VBA variables applying the various conventions in this section.

Table B.11 VBA Variable Name Examples

Declaration	Description
Dim oxlsaBudget **As Object**	Excel.Application
Function lci_ArraySum (ParamArray _ ravarNum() **As Variant**) **As Double**	Company identifier
Public giaintPartNum **As Integer**	Global index into array
Const clngCustNumMax **As Long** = 10000	**Const** for maximum value of CustID
Function FileLock(**ByVal** vstrFile **As** _ **String**) **As Integer**	**ByVal** argument

Creating Your Own Tags

What do you do when LNC doesn't address a particular object naming need?

- First, contact Kwery Corporation with an explanation of your need, so that the LNC style can be improved for the benefit of all users.
- Second, consider if what you are trying to do is covered by the style in some other way. For example, in your development team you call tables that link two other tables in a many-to-many relationship linking tables, and you want to create a new table tag tlnk as a result. However, on examination of all table tags, you would find `trel` already exists, defined as "Table (many-to-many relation)," which is the correct tag for what you need. Even though the nomenclature is not exactly what you require, it is better to use an existing tag than create another one.

When other options are exhausted, you can create a custom tag to address your need. When creating a custom tag, these should be your guidelines:

- Do not redefine an existing tag. No matter how badly you really want the three- or four-character combination for your own purpose, never reuse a defined tag.
- Do not change the rule for tags. Stay within the three- to four-character range followed by LNC.
- Use the conventions in existing tags as your guide for the new one. For example, all table tags start with `t`, all query tags with `q`, and so forth. Any new tags you make for these objects should begin with the correct letter. See Table B.12 for guidelines on standard tag components.

Oracle Ex. 1009, p. 1234
Oracle v. DeBellis
IPR2024-00221

Note

Some of the examples are from the Microsoft Office version of LNC. Tag components that can be easily inferred from the preceding tags are not listed in the table (for example, the component `fox` for FoxPro can be inferred from the tags `tfox` and `dfox`.)

When creating a new tag, it should be mnemonic enough to uniquely shorten the word it represents, and should only use characters from the root word or a generally accepted shorthand.

Table B.12 Some Standard Tag Components

Item	Segment	Examples	Location
bar	`br`	`mmbr, pbr, tbr`	anywhere
data/databound	`d`	`dout`	leading
database	`db`	`dbe`	leading
form	`f`	`fdlg`	leading
macro	`m`	`mmnu`	leading
MAPI	`mp`	`mpm`	leading
module	`b`	`bas`	leading
query	`q`	`qsel`	leading
report	`r`	`rdet`	leading
set	`st`	`rst`	anywhere
table	`t`	`tdf`	leading
view	`vw`	`lvw, tvw`	anywhere

To create tags for object variables pointing to (OLE) Automation server applications, start with the three-character DOS file extension for the files created by the server application, if unique and applicable. If not unique, create a meaningful abbreviation of the application name. Add to the three-character abbreviation a single character for the actual object that serves as the entry point for the application, such as `Basic` in `Word.Basic`.

For example, to create a tag for OLE Automation with Shapeware's Visio program, which is an OLE server, use either `vsd` (the data file extension) or `vis` (a mnemonic for Visio) as the basis for the tag, then add `a` for `Application`, because the entry point to Visio's automation engine is a call to `Visio.Application`. Thus the tag and its use in variable declarations would appear as:

```
Dim ovisa As Object
Dim ovisaDoc As Object
Set ovisa = CreateObject("Visio.Application")
Set ovisaDoc = ovisa.Documents.Open("C:\VISIO\HOUSE.VSD")
```


Oracle Ex. 1009, p. 1235
Oracle v. DeBellis
IPR2024-00221

How to Get the Complete Text of the LNCs

The full text of the Leszynski Naming Conventions for Microsoft Access and an expanded discussion of the naming convention philosophy appears in Que's *Access Expert Solutions* by Stan Leszynski. You also can obtain the full text of the Leszynski Naming Conventions for Microsoft Access, on which this abridged appendix is based, from Kwery Corporation. The following additional information is also available:

- The Leszynski Naming Conventions for Microsoft Solution Developers paper
- The Leszynski Naming Conventions for Microsoft Visual Basic paper
- Windows Help file versions of each LNC document
- LNC programmers' tools

Contact Kwery via the order line at 1-800-ATKWERY, or on the product information line at 206-644-7830. Kwery also can be reached on CompuServe at **71573,3261** or by fax at 206-644-8409. ●

Oracle Ex. 1009, p. 1236
Oracle v. DeBellis
IPR2024-00221

Appendix C

Data Dictionary for the Personnel Actions Table

You use the Personnel Actions table in examples in the following chapters:

4	"Working with Access Databases and Tables"
5	"Entering, Editing, and Validating Data in Tables"
9	"Understanding Operators and Expressions in Access"
10	"Creating Multitable and Crosstab Queries"
12	"Creating and Using Forms"
13	"Designing Custom Multitable Forms"
17	"Exporting Data to World Wide Web Pages"
20	"Adding Graphics to Forms and Reports"
24	"Working with Structured Query Language"

The step-by-step procedures for creating the Personnel Actions table and adding the first nine records to the table are included in Chapter 4, "Working with Access Databases and Tables."

Tables C.1 through C.5 provide a tabular data dictionary for the Personnel Actions table.

Table C.1 lists the values of the Field Name, Caption, Data Type, Field Size, and Format properties of the Personnel Actions table. These values are entered in the Properties text boxes for each field.

Oracle Ex. 1009, p. 1237
Oracle v. DeBellis
IPR2024-00221

Table C.1 Field Properties for the Personnel Actions Table

Field Name	Caption	Data Type	Field Size	Format
paID	ID	Number	Long Integer	General Number
paType	Type	Text	1	@> (all caps)
paInitiatedBy	Initiated By	Number	Integer	General Number
paScheduledDate	Scheduled	Date/Time	N/A	Short Date
paApprovedBy	Approved By	Number	Integer	General Number
paEffectiveDate	Effective	Date/Time	N/A	Short Date
paRating	Rating	Number	Integer	General Number
paAmount	Amount	Currency	N/A	##,##0.00#
paComments	Comments	Memo	N/A	None

Table C.2 lists the entries you make to assign default values to each field for which default values are required. You enter these values in the Default Values text box of the indicated field.

Table C.2 Default Field Values for the Personnel Actions Table

Field Name	Default Value	Comments
paType	Q	Quarterly performance reviews are the most common personnel action.
paScheduledDate	=Date()	The expression to enter today's (DOS) date.
paEffectiveDate	=Date() +28	Four weeks from today's date.

Table C.3 lists the values you enter as Validation Rules and the accompanying Validation text that is displayed in the status bar if an entry violates one of the rules. Only those fields with validation rules are shown in Table C.3.

Table C.3 Validation Criteria for Fields of the Personnel Actions Table

Field Name	Validation Rule	Validation Text
paID	>0	Please enter a valid employee ID number.
paType	"H" Or "S" Or "Q" or "C" Or "Y" Or "B"	Only H, S, Q, Y, B, and C codes may be entered.
paInitiatedBy	>0	Please enter a valid supervisor ID number.
paScheduledDate	Between Date()—3650 And Date()+365more	Scheduled dates cannot be than 10 years ago nor more than one year from now.

Oracle Ex. 1009, p. 1238
Oracle v. DeBellis
IPR2024-00221

Field Name	Validation Rule	Validation Text
paApprovedBy	`>0 Or Is Null`	Please enter a valid manager ID number or leave blank if not approved.
paRating	`Between 0 And 9 or Is Null`	Rating range is 0 for terminated employees, 1 to 9, or blank.

Table C.4 lists the key fields, indexes, and relationships for the Personnel Actions table. A composite key field is used so that duplication of an entry of a record for an employee is precluded. The default index that Access creates on the primary key field is shown for completeness; you do not add this index because Access creates indexes on key fields automatically. You establish the relationship with the Employees table by opening the Database window, choosing Tools, Relationships, and making the selections that are listed for the Relationships property in Table C.4 in the Relationships dialog.

Table C.4 Key Fields, Indexes, and Relationships for the Personnel Actions Table

Property	Value
Primary Key Fields	`paID;paType;paScheduledDate`
Primary Key Index	`paID;paType;paScheduledDate`
Relationships	Primary table: `Employees` Related table: `Personnel Actions` Enforce Referential Integrity: `True` (checked) Cascade Deletions: `True` (checked) Cascade Updates: `True` (checked)

Table C.5 lists the first nine entries in the Personnel Actions table that are used to demonstrate use of the table. The Scheduled and Effective entries are based on the HiredDate information in the Employees table. Enter these values after you have created the composite primary key for the table. Assigning performance ratings (except 0, terminated) to the records is optional.

Table C.5 First Nine Entries for the Personnel Actions Table

ID	Type	Initiated By	Scheduled	By	Approved Date	Effective Amount	New Comments
1	H	1	01-May-92		01-May-92	2,000	Hired
2	H	1	14-Aug-92		14-Aug-92	3,500	Hired
3	H	1	01-Apr-92		01-Apr-92	2,250	Hired
4	H	2	03-May-93	2	03-May-93	2,250	Hired
5	H	2	17-Oct-93	2	17-Oct-93	2,500	Hired
6	H	5	17-Oct-93	2	17-Oct-93	4,000	Hired
7	H	5	02-Jan-94	2	02-Jan-94	3,000	Hired
8	H	2	05-Mar-94	2	05-Mar-94	2,500	Hired
9	H	5	15-Nov-94	2	15-Nov-94	3,000	Hired

Oracle Ex. 1009, p. 1239
Oracle v. DeBellis
IPR2024-00221

Index

Symbols

(date specifier) reserved word (Jet SQL), 842
& (ampersand) operator, 171
(>) greater than operator, 291
(>=) greater than or equal to, 291
* (multiplication) operator, 171, 290
+ (plus sign) operator, 171, 290
- (minus sign) operator, 171, 290
/ (division) operator, 171, 290
< (less than) operator, 171, 291
<= (Less than or equal to) operator, 291
<> (not equal) reserved word (Jet SQL), 842
> (greater than) operator, 171
? (LIKE wild card) reserved word (Jet SQL), 842
1 Database command (Window menu), 408, 434, 524
16-bit OLE Controls, converting 32-bit ActiveX Controls, 1159
16-bit programmable objects, interacting
  Windows 95, 1099-1100
  Windows NT, 1099-1100
16-bit Unicode, 61-62
32-bit ActiveX Controls, 16-bit OLE Controls, converting, 1159
32-bit applications, run-time access versus retail access, 1136-1137
32-bit OLE, 665
  ActiveX Controls, 682
  Automation, 681-682
  compliant applications, 668-670
  .ocxs file, 31
32-bit programmable objects, interacting
  Windows 95, 1099-1100
  Windows NT, 1099-1100

A

About dialog box, 91
accelerator keys, 1163
Access
  implementing with object controls, hyperlinks, 595-596
  installing to networks, 866-867
  Microsoft development, 1095-1096
  prior versions, event-handling capabilities, 1049
  run-time Access, distributing, 867
Access 1.0
  Access 95, converting .MDF files, 1155
  conversion errors, 1156-1160
Access 2.0
  Access 95
    converting .mdb files, 1155
    enhancements, 29-32
  Access 97
    converting .mdb files, 1154-1155
    executing applications, 1152-1154
    migrating .mdb files, 1152-1154
  Calendar Control, 1124-1131
  conversion errors, 1156-1160
  custom libraries
    32-bit conversion, 1157-1159
    converting, 1157-1159
  macros, converting to VBA code, 1080-1086
  tables, converting, 1159

Oracle Ex. 1009, p. 1240
Oracle v. DeBellis
IPR2024-00221

Access 95
  Access 2.0, enhancements, 29-32
  Access 97, compared, 22-29
  conversion errors, 1156-1160
  converting
    Access 1.0 .MDF files, 1155
    Access 2.0 .MDF files, 1155
  databases, converting, 1159-1160
  image controls, 707-709
  macros versus code, 977-978
  versus Access 97, disk space, 114
Access 97
  Access 95 databases, converting, 1159-1160
  Access modules, 985-986
  ActiveX Controls, registering, 1123-1124
  AutoExec macros, custom CommandBar, 1079
  automation servers, implementing, 1131-1133
  Calendar Control, programming, 1127-1131
  Chart Wizard, previous versions, 710-711
  constants, naming, 1225
  conversion errors, 1156-1160
  converting Access 2.0 .MDB files, 1154-1155
  converting format, databases, 104
  creating
    expressions, 309-319
    ODBC data source, 637-642
  data type conversion functions, 304
  database applications, characterisitics, 109-110
  database elements
    defining, 107-110
    forms, 107-110
    macros, 107-110
    modules, 107-110
    queries, 107-110
    reports, 107-110
    tables, 107-110
Database toolbar, 52
database utilities, 102-105
databases
  .mdb extension, 111
  disk space, 114
  opening, 48-51
  repairing, 104-105
  windows, 51-52
Datasheet View options, setting, 88-89
default options, setting, 81-89
DoCmd methods, implementing, 1068-1070
document windows
  maximizing, 69-70
  minimizing, 70-71
executing Access 2.0 applications, 1152-1154
Expression Builder, query criteria, 314-316
external databases, querying, 367-369
financial functions, 306
forms, exporting to ASP files, 660-662
function keys, 79-81
  fields, grids and text boxes, 80
  global, 79
  module window, 80-81
functions, 294-295
  closed queries, 66
  macros, 65-66
  modules, 65-66
  open queries, 66
  printing, 66
  security, 66
graphic images, scaling, 696-699
Help menu options, 91
help system
  context-sensitive help, 90-91
  overview, 89-102
Help Topics dialog box, 91-97
HTML templates, 606-608
hyperlink field data type, 585
identifiers, 294
image controls, 707-709
implementing, automation servers, 1100-1102
importing HTML tables, 618-623
indexes per table limit, 1159
installing, 32-42
Internet, 22
intrinsic constants, 309
launching, 48-50
Leszynski Naming Conventions, 1208-1209
  user levels, 1209
library references, storing, 1158-1159
literals, 293
macros, converting to VBA code, 1080-1086
macros versus code, 977-978
Mail Merge Wizard, 763-764
menu macros, 1218-1219
migrating, Access 2.0 .MDB files, 1152-1154
miscellaneous functions, 307-308
named constants, 309
new features, 22-29
Object Browser, implementing, 1027-1028
object collections, 1023-1024
object libraries
  Microsoft Access 97, 1026-1027
  Microsoft DAO 3.5, 1026-1027
  Microsoft Office 8.0, 1026-1027
  Visual Basic for Applications 5.0, 1025-1027
ODBCDirect, 1023
Office 97 Developer Edition, 1135-1136
Office Assistant, 97-102
operating modes
  Design, 66-67
  Run, 66-67
  Startup, 66-67

Oracle Ex. 1009, p. 1241
Oracle v. DeBellis
IPR2024-00221

operators, 289-293
print formats, 61-62
reports, exporting, 610-613
Setup Wizard, 1138-1147
shortcut menus via mouse button, 78-79
spreadsheet data conversion, tables, 735-737
SQL statements, processing, 284-286
system default options, setting, 82-88
System.mdw, database master file, 112
Table view
components, 67-69
toolbars, 71-75
toolbars, customizing, 75-78
variant data types, 298-300
VBA references, 985-986
Access 97 Options dialog box, 646
Access command (Help menu), 91
access modules (Access VBA), 981-985
Access SQL, 1163
SELECT * FROM table name, 350
access tokens, 1163
Access VBA
arrays, declaring, 992-993
breakpoints, adding, 1015-1016
comparisons with
dBASE, 979-980
PAL, 979-980
Visual VBA, 979-980
xBase, 979-980
Compiler, 1012-1014
control properties, 1088-1089
conventions
naming, 980-981
typographic, 980-981
data events, responding, 1090-1094
database object data types, 986-988
date/time functions, 300-301
developmental history, 978-979
dynamic arrays, creating, 992
Err function, error detection, 1006-1007
Excel worksheets, creating, 1112-1115
field data types, 986-988
functions
executing, 984-985
versus subprocedures, 984-985
Help System, 1011-1012
hierarchy, data access objects, 1029-1030
implementing, 980
intrinsic symbolic constants, 996-997
Jet 3.5 database engine, 1023-1024
looping operations
Do Until...Loop, 1003
Do While...Loop, 1003
For...Next statement, 1002
module shortcut keys, 1010
module window, toolbar, 1008-1010
modules
access, 981-985
form, 981-985
report, 981-985
text comparisons, 1020
multidimensional arrays, creating, 992-993
named arguments, 997
object properties, With...End structure, 994
objects, referencing, 985-986
open forms, referring, 1087-1088
optional arguments, 997
procedures, 983-985
invoking, 983
program flow
conditional execution, 998
repetition, 998
termination, 998
programming advice, 979-980
report forms, referring, 1087-1088
run-time errors, handling, 1005-1007
source codes, creating Excel workbook objects, 1113-1114
subform controls, 1089-1094
symbolic constants, 995-997
system-defined symbolic constants, 996-997
user-defined functions, creating, 1020-1022
variables, 988-995
explicit, 989
implicit, 988-989
named database objects, 993
naming conventions, 994-995
scope level, 990-991
user-defined data types, 991-992
Win32 API Viewer, prototype declarations, 1147-1150
accessing fields in tables, 349-351
ACLs (Access Control Lists), 1163
action queries, 279-280
append, 371
cascading deletions, 388
cascading updates, 388
creating, 280-282
delete, 371
make-table, 371
overview, 371
parameters, adding, 282-284
records, deleting, 381-383
syntax, 860-861
tables, appending records, 378-380
update, 371
activating Office Assistant, 98-102
activations (OLE), 1163
active applications, 1163
Active Server Pages, *see* **ASP**
active windows, 1163
ActiveMovie, 1163
Microsoft Internet technologies, 581

Oracle Ex. 1009, p. 1242
Oracle v. DeBellis
IPR2024-00221

ActiveMovie Streaming Format (ASF), 579
ActiveX, 665, 1163
automation, 1166
components, 1164
data objects, 1164
documents, 1164
Microsoft Internet technologies, 581
scripting, 1164
server framework, 1164
technologies, 581
ActiveX Controls, 665, 669, 682, 1164
applications, adding, 1122-1123
Calendar Control, 1124-1131
programming, 1127-1131
registering, 1123-1124
ActiveX Control command (Insert menu), 1124
ActiveX Controls command (Tools menu), 1011, 1123
ActiveX Data Objects (ADO), 652, 1096-1097
ActiveX Web site, 1096
Add ActiveX Control dialog box, 1123
Add Reference dialog box, 1157
Add Tables dialog box, 263
Add Watch command (Debug menu), 1011
Add-In Manager, 947
add-ins, 30, 1164
Database Documentor, 820-823
Database Splitter, 30, 874
Import Database, 894
Linked Table Manager, 218-219, 874, 908
SQL Server Browser, 947, 951-953
Add-Ins command (Tools menu), 112, 218
Add/Remove Programs option (Control Panel), 921
adding
Access VBA breakpoints, 1015-1016
ActiveX Controls, applications, 1122-1123
bound object controls, forms, 692-701
bound object frames, graphics images, 698-699
calculated controls, reports, 534-539
Calendar Control forms, 1124-1127
combo boxes, forms, 474
controls in forms, 444-446, 451-495
data sources (SQL Server), 924-935
event-handling code, Command Button Wizard, 1056
fields, asterisk, 350
fixed value lookup lists in tables, 337-340
footers in forms, 510
groups, 890-891
headers in forms, 510
indexes in tables, 154-155
linked subreports within reports, 564-568
lookup fields
foreign-key drop-down list, 333-337
tables, 332-333
make-table queries, parameters, 282-284
option buttons, forms, 464-472
queries, parameters, 359-360
query criteria (Expression Builder), 314-316
records in tables, 168-170, 175-177
reports
footers, 557-559
headers, 557-559
subreports, unbound reports, 568-569
tables
existing databases, 137-138
type field value, 742-743
toolbox controls within reports, 569-570
unbound object frames in forms, 701-707
AddMenu macro actions, replacing, 1075-1078
AddMenu method, 1070-1075
addresses, 1164
IP addresses, 1184
administrating databases, 910-912
administrators, 884
databases, 1174
names, establishing, 884
Adobe Photoshop, 700
ADT (Access Developer's Toolkit), 21, 1164
advanced filters, records, 193
advanced options
DDE linking, 86-88
ODBC, 86-88
OLE operations, 86-88
setting system default options, 86-88
advanced sorts in records, 193
aggregate functions, 1164
domain aggregate functions, 1177
SQL, 351-352
aggregate objects, 1164
aliases, 1164
aligning form controls, 424
grids, 422-423
allow zero length, table field properties, 118-119
Alt+Backspace key combination, 164-166
altering table fields, 155
Analyze command (Tools menu), 813
And operator, 171
Annotate command (Help menu), 97
ANSI (American National Standards Institute), 234, 1165
character set, 61-62
Jet SQL, compared, 838-847

Oracle Ex. 1009, p. 1243
Oracle v. DeBellis
IPR2024-00221

SQL, 836
keyword substitutes, 840
keywords, 840
non-supported keywords, 846-849
ANSI SQL (ANSI Structured Query Language), reserved words
AUTHORIZATION, 847
BEGIN r, 847
CHECK r, 847
CLOSE r, 847
COMMIT r, 847
CREATE VIEW r, 847
CURRENT r, 847
CURSOR r, 847
DECLARE r, 847
DROP VIEW r, 847
FETCH r, 847
GRANT r, 847
PRIVILEGES r, 847
REVOKE r, 847
ROLLBACK r, 847
TRANSACTION r, 847
VALUES r, 847
WORK r, 847
APIs (Application Programming Interfaces), 1165
MAPI (Messaging API), 1186
ODBC, 917
Win32 API, 1205
Append Query command (Query menu), 379
appending
records
query properties, 346-347
tables, 170, 378-380, 433-436
text in tables, 245
Apple, OpenDoc, 1191
applets, 1165
Applicable Filter dialog box, 198
Application Close button, 1165
Application Control-menu box, 1165
application databases, 798
application objects, 873
applications, 1165
active, 1163
adding ActiveX Controls, 1122-1123
automation-compliant applications, 669
compliant applications, 32-bit OLE, 668-670
contact management applications, creating, 958-962
container applications, 668
database categories, 396
full-server applications, 669
launching, 1185
mini-server applications, 669
multithreaded, 1188
multiuser applications, 868-877
opening, 877-881
securing, 881-892
networking applications, 865-866
object-enabled applications, 671
OLE Registry, 686-687
scalable applications, creating, 913-914
server-only applications, 668
upsizing (Upsizing Wizard), 947-951
Apply Filter button, 187, 198
ApplyFilter method, 1071-1075
argument method, object collections, 1030
arguments, 1165
DoCmd methods, 1070-1075
arithmetic operators, 171, 289-290
structure, 289-290
arrays, 1166
assigning values, For...Next statements, 1003
controls, 1172
declaring (Access VBA), 992-993
dynamic, 992-993
multidimensional, 992-993
scope levels, 993
Arrow Key Behavior property, 163
AS keyword (SQL), 853
ASCII (American Standard Code for Information Interchange), 234, 1166
ASP (Active Server Pages)
ActiveX Data Objects (ADOs), 652
exporting
forms, 660-662
queries, 655-660
features, 652
\InetPub subfolders, creating, 652-655
Microsoft Internet technologies, 582
server-side scripting, 652
source codes, HTML .asp file, 658-660
assigning
fields function keys, 80
For...Next statements, array values, 1003
global function keys, 79
grids function keys, 80
module window function keys, 80-81
Personnel Action Table, field information, 140-151
text box function keys, 80
variables, 1166
assignment operators, 290-291
asterisks (*), adding fields, 350
Asymetrix Corporation, InfoModeler, 801
asynchronous, 1166
attached tables, 1166
attribute
data types, 795
domains, 795
values, 795
attributes
data entities, objects, 791-792
databases, 794
tables, viewing, 57-59
authentication, 1166
AUTHORIZATION reserved word (ANSI SQL), 847
Auto List Members, 26

Oracle Ex. 1009, p. 1244
Oracle v. DeBellis
IPR2024-00221

Auto Quick Info, 26
AutoExec macros, custom CommandBar, replacing, 1079
AutoFormat command (Format menu), 407, 524
AutoFormats
  creating, 412-413
  customizing, 412-413
  deleting, 412-413
  forms, 411-413
  reports, customizing, 526
automated printing (Debug window), 1018-1019
automatically finding record matches, 186
automation, 1166
automation clients, 1166
automation components (ActiveX), 1097-1099
Automation OLE, 681-682
automation servers, 1166
  Access 97, implementing, 1100-1102, 1131-1133
  hardware requirements, 1099
  RAM requirements, 1099
  Registry entries, manipulating, 1108-1109
automation-compliant applications, 669
AutoNumber field, modifying tables, 739-742
AutoNumbers, Increment AutoNumbers, 1166
AutoPlay, 1166
avoiding
  infinite loops, 1004-1005
  "spaghetti code," 999

B

background multitasking, 1167
background colors, forms, creating, 414
backing up databases, 910-912
Backspace key, 164
backups, 1166
  creating in Registry, 687-689
base dates, 1167
base names
  compound objects, 1214
  creating
    objects, 1212-1215
    VBA objects, 1219
  field objects, 1214
  length limits
    objects, 1213
    VBA objects, 1219-1220
  naming rules, 1212-1213
  naming VBA objects, 1219
  objects, 1211-1212
  ordering objects, 1214-1215
  qualifiers, objects, 1211-1212
  suffixes in objects, 1212
base tables (relational databases), 796, 1167
BASIC (Beginner's All-Purpose Symbolic Instruction Code)
  conditional statements
    If...Then...End If, 999
    Select Case...End Select constructs, 1000-1002
  developmental history, 978
  Go To statements, 998
  program flow, branching, 998
  string functions, text manipulation, 302-304
batches, 1167
BDCs (Backup Domain Controllers), 1167
Beep method, 1071-1075
BEGIN reserved word (ANSI SQL), 847
Beginner's All-Purpose Symbolic Instruction Code, *see* BASIC
Best Fit button, 200
Between operator, 171
binary files, 1167
BINARY reserved word (Jet SQL), 841
binary strings, 1167
binary trees, indexes, 826-827
binding, 1167
  controls in forms, 464-472
  reconciliation handlers, 956
bit, 1167
bitmap images
  file formats, 699-701
    .bmp files, 699
    .dib files, 699
    .gif files, 700
    .jpg files, 699
    .pcd files, 700
    .pcx files, 699
    .png files, 700
    .tga files, 700
    .tif files, 699
bitmapped images, linking, 679
bitmaps, 1167
bitwise, 1167
blank forms, creating, 454-455
blank reports, creating (Report Wizard), 555-564
blitting, 1167
.bmp files, 699, 700
bookmarks, linking (Word 97), 588-590
Boolean, 1167
BOOLEAN reserved word (Jet SQL), 841
borders
  form controls, changing, 424-425
  forms, changing, 417-421
bound controls (forms), 444
bound object controls
  displaying (Personnel Actions), 693-696
  forms, adding, 692-701
  troubleshooting, 693
Bound Object Frame Properties window
  Clip, 696-699
  Stretch, 696-699
  Zoom, 696-699
bound object frames
  graphics images, adding, 698-699
  toolbox controls, 569-570
Bound Object tool (Toolbox), 446
bound text boxes, creating, 458-463
branching program flow (BASIC), 998

Oracle Ex. 1009, p. 1245
Oracle v. DeBellis
IPR2024-00221

breakpoints, 1168
adding (Access VBA), 1015-1016
breaks, 1168
BRI (Basic Rate Interface), 1168
Briefcase
GUIDs (Globally Unique IDs), 956-957
replicating contact management applications, 958-962
tracking contents, 956
Briefcase menu commands, Update All, 957, 970
Briefcases
moving, 967
replicating, 956-957, 1168
replica sets, 962-965
synchronizing, 965-966
updating, 957-966
buffers, 1168
builders, 447, 1168
Expression Builder, 447
Query Builder, 447
building
merge document records, 764-767
select queries, 372-373
built-in functions, 1168
business rules, 1168
buttons
Application Close, 1165
Apply Filter, 187, 198
Best Fit, 200
Change Owner, 893-894
Clear Grid, 191
Close Window, 174
command button, 1171
Datasheet View, 831
Design View, 172
Filter by Form, 190
Find First, 184
First Record, 187
New Database, 207
New Record, 169
option button, 1191
Paste, 170
Properties, 173
Tentative Append Record, 677
toolbars
changing appearance, 449
creating, 449
removing, 449
Report Print Preview mode, 525
BYTE reserved word (Jet SQL), 841

C

caches, 1168
calculated controls, reports, adding, 534-539
calculated controls (forms), 444
calculated text boxes, creating, 458-463
calculating expressions, query field values, 316-319
calculations
selected table records, executing, 356-359
table records, executing, 352-356
Calendar, 31
Calendar Control
forms, adding, 1124-1127
programming, 1127-1131
Call Stack command (View menu), 1009
CancelEvent method, 1071-1075
candidate fields (Personnel Actions Table), 140-151
candidate keys in relational databases, 796
captions, 1168
tables, field properties, 117-118
captions, 1168
carets, 1169
Cartesian product, 1169
cascading deletions, 1169
action queries, 388
relationships, establishing, 388-389
tables, 151-152
testing, 390
cascading updates
action queries, 388
relationships, establishing, 388-389
tables, 151-152
testing, 390-391
CASE (Computer-Aided Software Engineering) tools, 801-802
case expressions, Select Case...End Select construct, 1000-1002
case sensitivity, 1169
categories
creating reports, 559-564
databases, grouping, 552-553
Categories table, 52-53
graphics
editing, 56-57
viewing, 56-57
CBF (code-behind-forms), 29
CCL (Cursor Control Language), 839
CDFS (CD-ROM file system), 1169
cells
linking Excel 97, 760-762
specifying range (Customer.xls), 1111
tables
editing, 53-56
selecting, 53-56
viewing, 52-53
Cells command (Format menu), 75, 200
Cells Effects dialog box, 200
Change Access Rights dialog box, 905
Change Folder dialog box, 34
Change Owner button, 893-894
Change To command (Format menu), 495
channels, 1169
characters, selecting (Office Assistant), 100
Chart menu commands, Chart Type, 717
Chart Type command (Chart menu), 717
Chart Type dialog box, 717

Oracle Ex. 1009, p. 1246
Oracle v. DeBellis
IPR2024-00221

Chart Wizard, 447
creating unlinked graphs, 711-715
crosstab queries, 721-723
earlier versions versus newer versions, 710-711
charts
columnar data, troubleshooting, 723
creating
Microsoft Graph 97, 709-727
queries, 709-711
Check Box tool (Toolbox), 445
check boxes, 1169
forms, 464-472
toolbox controls, 570
CHECK reserved word (ANSI SQL), 847
Checking and Reporting Errors dialog box, 776
child windows, 1169
Choose Columns dialog box, 773
Choose Data Source dialog box, 771
Choose Profile dialog box, 571
chunks, 1169
class module events, 1062-1068
class modules, 29
Object Browser, viewing, 1054-1056
overview, 1049-1050
classes, objects, 672-673
clauses (SQL), 1169
Clear All Breakpoints command
Debug menu, 1011
Run menu, 1017
Clear Grid button, 191
CLI (Call Level Interface), 919
client/server databases, 913
connecting to, 915
environment, 915
ODBC (Open Database Connectivity), 917-918
drivers, 918-920
installing drivers, 920-923
SQL Server, 915
installing, 920-923
tables, exporting, 945-953
clients, 1169
automation clients, 1166
clip option, Bound Object Frame Properties window, 696-699
Clipboard, 666, 1170
controls, copying, 463
form controls
copying, 472-474
deleting, 424
keyboard combinations, 165-166
query data, transferring, 276-278
records
exporting, 253-257
importing, 247-250
replacing, 250-251
clipping graphic images, 698-699
Close command (File menu), 51, 162
Close method, 1071-1075
CLOSE reserved word (ANSI SQL), 847
Close Window button, 174
closed queries, Access 97 functions, 66
closing worksheet objects (Excel 97), 1111-1112
CLSIDs (class identifiers), 1170
clustered indexes, 827-828, 1170
clustering servers, 1170
code, 1170
compiling, 1171
executable code, 1179
invoking, 1184
machine languages, 1186
objects, 1190
Code command (View menu), 408, 524, 982
code templates, 1170
Code window, 1170
coercion, 1170
collections of objects, 1170
color palette, 1170
colors
foreground colors in forms, 417-421
forms, changing, 413-421
objects, changing, 413-421
RGB (red, green, blue) color scheme, 1196
column headers
queries, modifying, 271-273
renaming, 272-273
columnar data, troubleshooting, 723
columns, dynasets, 1178
COM (Component Object Model), 665, 1171
Combo Box tool (Toolbox), 445
Combo Box Wizard, 475
records searches, creating, 486-489
combo boxes
forms, 474-486
adding, 474
list boxes, compared, 474
populating (Query Builder), 484-490
records searches, 486-489
static value lists, creating, 483-486
combo lists, 1171
comma-delimited text files, 243-257
command button, 1171
Command Button tool (Toolbox), 445
Command Button Wizard, adding event-handling code, 1056
Command Button Wizard dialog box, 1057
command buttons in toolbox controls, 570
command-line options, shared databases, opening, 877-881
CommandBar objects, 23
references to Microsoft Office 8.0 Object Library, 1075-1078
replacing AddMenu macro actions, 1075-1078

Oracle Ex. 1009, p. 1247
Oracle v. DeBellis
IPR2024-00221

commands
Briefcase menu, Update All, 957, 970
Chart menu, Chart Type, 717
Data menu, Series in Rows, 723
Debug menu
Add Watch, 1011
Clear All Breakpoints, 1011
Compile All Modules, 1043
Compile Loaded Modules, 1009
Edit Watch, 1011
Run to Cursor, 1011
Show Next Statement, 1011
Step Into, 1009
Step Over, 1009
Toggle Breakpoint, 1009
Edit menu
Copy, 73, 165, 407, 433, 524
Cut, 72, 165, 170, 407, 433, 524
Delete Record, 74, 434
Document Object, 783
Find, 84, 183, 433
Go To, 434
Paste, 73, 160, 170, 407, 433, 524
Paste Append, 170
Paste Special, 678
Redo, 1009
Rename, 279, 940
Replace, 84, 186
Select All, 540
Select All Records, 276, 379
Select Form, 420, 423, 428
Select Report, 536, 559
Undo, 73, 160, 407, 433, 524, 1009
Worksheet Object, 758
File menu
Close, 51, 162
Exit, 56, 676
Exit & Return, 705
Export, 945
Export to HTML, 611
Get External Data, 207, 217, 618, 732, 892, 1156
Import, 618
Load Database File, 1148
Load from Query, 198
New, 113, 160
New Database, 207, 732
Open, 163
Open Database, 50, 206
Page Setup, 60, 275, 544, 783
Print, 60, 72, 433, 544
Print Preview, 59, 72, 407, 433, 523
Properties, 668
Return Data to Microsoft Word, 776
Save, 72, 79, 407, 433, 523
Save As, 279
Save As/Export, 597, 655
Save As HTML, 614, 643
Save As Query, 197
Save Copy As, 705
Send, 570
Undo, 54
Format menu
AutoFormat, 407, 524
Cells, 75, 200
Change To, 495
Font, 75, 199
Freeze Columns, 180
Hide Columns, 199
Row Height, 200
Selected Axis, 717
Selected Chart Title, 716
Size, 422
Unfreeze All Columns, 183
Unhide Columns, 199
Help menu
Access, 91
Annotate, 97
Contents and Index, 91, 297
Keep Help, 97
Microsoft Access, 408
Microsoft Access Help, 98, 434
Microsoft on the Web, 91
Print Topic, 97
Use System, 97
What's This?, 90
Hyperlink menu, Edit Hyperlink, 586
Insert menu
ActiveX Control, 1124
Date and Time, 765
Hyperlink, 120, 407, 433, 524
Insert Hyperlinks, 73
Module, 1008
Name, 222, 1103
Object, 703
Procedure, 1021
Programs menu
Start, 48
Query menu
Append Query, 379
Crosstab Query, 362
Delete Query, 382
Make-Table Query, 374, 737, 743
Parameters, 360
Show Table, 744
SQL-Specific, 941
Update Query, 386, 742
Records menu
Filter, 73, 187-199, 433
Go To New, 74
Refresh, 442
Remove Filter/Sort, 197, 248
Save Record, 169, 435
Sort, 73, 181, 433
Run menu
Clear All Breakpoints, 1017
End, 1009
Reset, 1009
Start menu
Documents, 668
Find, 940
Run, 32
Table menu, Convert Table to Text, 276
Tools menu
ActiveX Controls, 1011, 1123
Add-Ins, 218

Oracle Ex. 1009, p. 1248
Oracle v. DeBellis
IPR2024-00221

Add-Ins, 112
Analyze, 814
Compact Database, 103
Convert, 1155
Convert Form's Macros to Visual Basic, 1082
Database Utilities, 66, 102, 162, 1155
Instant Watch, 1009
Macro, 1082, 1131
Make MDE File, 105
Merge It with MS Word, 768
Object Browser, 1009
Office Links, 201, 525, 768
Options, 81, 112, 167, 410, 872
References, 112, 985, 1011, 1131
Relationships, 148, 159, 376
Replication, 957
Security, 66, 883
Spelling, 72, 407, 433, 524
Visual Basic Editor, 1131
View menu
Call Stack, 1009
Code, 408, 524, 982
Datasheet, 53, 716
Datasheet View, 72, 146
Debug Window, 295, 1009
Design View, 72, 146, 433
Details, 115, 969
Field List, 407, 524
Form, 407
Form Header/Footer, 406, 454
Grid, 423
Image, 696
Indexes, 58, 215
Join Properties, 341
Object Browser, 985, 1132
Page Header/Footer, 557
Pages, 525
Procedure Definition, 1011
Properties, 57, 137, 408, 524
Report Header/Footer, 557
Sorting and Grouping, 524
Source, 600
SQL, 198, 357, 753
SQL View, 843
Table Design, 57
Table Names, 453
Toolbars, 71, 448
Toolbox, 407, 444
Totals, 313
Zoom, 525
Window menu
1 Database, 408, 434, 524
Database, 206
commas (SQL), 849
comments, 1171
COMMIT reserved word (ANSI SQL), 847
common dialog boxes, 1171
Common Object Model (COM)
ActiveX technologies, 1096-1097
Microsoft Internet technologies, 580
OLE technologies, 1096-1097
specifications, obtaining, 580, 1096-1097
Compact Database command (Tools menu), 103
Compact Into dialog box, 162
compacting
database files, 911
databases, 102-104
comparison operators, 171
compilation results, displaying (Debug window), 1016-1017
Compile All Modules command (Debug menu), 1043
Compile Loaded Modules command (Debug menu), 1009
compilers, defined, 1012-1014
compiling code, 1171
Component Builder dialog box, 1142
components (ActiveX), 1164
composite
filters, 195-197
indexes, 1171
keys, 1171
relational databases, 797
menus, 1171
monikers, 1171
compound
base names, VBA objects, 1220
document, 1172
filters, 193-195
compounds, 1172
concatenation, 1172
operators, 291
conceptual data models, 798
concurrency, 1172
conditional execution, program flow (Access VBA), 998
conditional statements
If...Then...End If, 999
Select Case...End Select constructs, 1000-1002
configuring
forms, 427-429
keyboard, data entry, 167-168
subforms, 430-431
system files, workgroups, 877
Tab controls, 492
Confirmation dialog box, 127, 178
connect-string properties (SQL), setting, 942-944
connecting to client/server databases, 915
consignment operators, 290-291
constants
global, 1181
intrinsic, 309
named, 309
naming Access 97, 1225
unique prefixes, 1225
CONSTRAINT statement (SQL DDL), 846
contact management applications, creating, 958-962

Oracle Ex. 1009, p. 1249
Oracle v. DeBellis
IPR2024-00221

ContactData.mdb, replica sets, creating, 962-966
container applications, 668
containers (OLE), 673-681, 1172-1173
contents, tables, printing, 59-62
Contents and Index command (Help menu), 91, 297
Contents tab (Help Topics dialog box), 92
context-sensitive help
Access 97 help system, 90-91
obtaining, 90-91
context switching, 1172
continuous forms, creating, 503-509
control objects, 671
level one tags, 1217-1218
level two tags, 1217-1218
LNC, 1209
Control Panel
Add/Remove Programs option, 921
ODBC Administrator, 924-929
control properties (Access VBA), 1088-1089
Control Wizards tool (Toolbox), 445
controlling page breaks in reports, 559
controls, 1172
appearance
Form Design view, 450-451
Run mode, 450-451
arrays, 1172
custom controls, 1173
forms
adding, 444-446, 451-495
aligning, 424
aligning in grids, 422-423
binding, 464-472
categories, 444
changing borders, 424-425
changing types, 495
copying, 463
copying to other forms, 472-474
deleting, 424
moving, 421-424
selecting, 421-424
sizing, 421-422
Tab controls, 489-495
text controls, 425-426
reports
calculated controls, 534-539
deleting, 526-532
editing, 526-532
formatting, 540
relocating, 526-532
conversations, 1172
conversion errors
Access 95, 1156-1160
Access 97, 1156-1160
Convert command (Tools menu), 1155
Convert Form's Macros to Visual Basic command (Tools menu), 1082
Convert Table to Text command (Table menu), 276
converting
32-bit ActiveX Controls from 16-bit OLE controls, 1159
Access 1.0 .MDF files to Access 95, 1155
Access 2.0
custom libraries, 1157-1159
tables, 1159
Access 2.0 .MDB files to Access 97, 1154-1155
Access 2.0 .MDF files to Access 95, 1155
Access 95 databases, 1159-1160
databases to Access 97 format, 104
line graphs, properties, 717-720
macros (VBA code), 1080-1086
queries, 280-282
select queries, make-table queries, 374-375
spreadsheet data in tables, 735-737
Win16 function calls to Win32 API, 1156
Copy command (Edit menu), 73, 165, 407, 433, 524
Copy Progress dialog box, 40
copying
controls in forms, 463, 472-474
Northwind.mdb database, 162-163
records
filtered, 201-202
sorted, 201-202
tables to Windows Clipboard, 159-160
Corel PhotoPAINT, 700
correlated subqueries, 1172
counters, 1172
country selection for merge documents (Mail Merge Wizard), 765-767
Create Data Source dialog box, 771
Create New Data Source dialog box, 639, 930
Create New Data Source Wizard, 230
CREATE TABLE statement (SQL DDL), 846
CREATE VIEW reserved word (ANSI SQL), 847
creating
Access VBA
dynamic arrays, 992
Excel worksheets, 1112-1115
multidimensional arrays, 992-993
action queries, 280-282
ASP, \InetPub subfolders, 652-655
crosstab queries, 361-367
Customer.xls (Excel 97), 1102-1103
databases, 112-114
.mde files, 105
default values, expressions, 309-310
distribution disks
images, 1138-1150
Setup Wizard, 1138-1147

Oracle Ex. 1009, p. 1250
Oracle v. DeBellis
IPR2024-00221

expressions, 309-319
query criteria, 312-314
form letters, queries, 777-778
forms for documents, 781-784
graphs, crosstab queries, 721-723
High table, 743-746
HTML Export, Web pages, 643-651
IDC, Web pages, 627-637
indexes
multiple-field, 154-155
single-field, 154-155
input masks, 128-129
LNC tags, 1227-1228
logos for Personnel Actions Table, 702-705
Low table, 743-746
Mail Merge Wizard, form letters, 764-767
make-table queries, tables, 371-378, 737-739
merge documents from differing data source, 767-770
Microsoft Graph 97
charts, 709-727
graphs, 709-727
Microsoft Query, mail merge data source, 770-777
multi-field primary keys, 153
multicolumn equi-joins, 329-332
nilTestQuery function, 1042-1044
not equal theta joins in tables, 344-346
object variables, 1032-1033
object base names, 1212-1215
ODBC data source (Access 97), 637-642
OLE links, 760-762
outer joins in tables, 340-342
Personnel Actions Table, 141-144
primary keys in tables, 152-153
queries, 262-279
charts, 709-711
dynamic Web pages, 643-651
indirect table relationships, 326-329
query field values, 316-319
QueryDef objects, 1040-1041
references
data access objects, 1025-1027
object collections, 1030
Report Wizard, blank reports, 555-564
reports by category, 559-564
self-joins in tables, 342-344
single column equi-joins, 323-326
tables
Datasheet view, 134-137
Table Wizard, 130-134
Web pages, 601-603
unbound reports, multiple subreports, 568-569
unlinked graphs (Chart Wizard), 711-715
user-defined functions, 1020-1022
VBA objects, base names, 1219
Volume table, 743-746
Web pages
queries, 603-606
unformatted, 601-603
Web Publishing Wizard
home pages, 616-618
static Web sites, 614-618
creating Excel workbook objects source code, 1113-1114
crosstab queries, 279-280
creating, 361-367
graphs, 721-723
detail level, modifying, 365-367
fixed-column headings, implementing, 363-365
Crosstab Query command (Query menu), 362
Crosstab Query Wizard, queries, creating, 361-362
Ctrl+' (apostrophe) key combination, 167
Ctrl+" (quote) key combination, 167
Ctrl++ (plus) key combination, 167
Ctrl+- (minus) key combination, 167
Ctrl+: (colon) key combination, 167
Ctrl+; (semicolon) key combination, 167
Ctrl+C key combination, 166
Ctrl+End key combination, 163
Ctrl+Enter key combination, 167
Ctrl+Home key combination, 164
Ctrl+Insert key combination, 166
Ctrl+V key combination, 166
Ctrl+X key combination, 166
Ctrl+Z key combination, 164-166
CTS (Carpal Tunnel Syndrome), 162
CUA (Common User Access), 1172
CURRENCY reserved word (Jet SQL), 841
current databases, 1173
current records, 1173
CURRENT reserved word (ANSI SQL), 847
current statements, 1173
CURSOR reserved word (ANSI SQL), 847
Cursor Stops at First/Last keyboard option, 168
custom CommandBar
replacing AutoExec macros, 1079
startup properties, specifying, 1079
custom controls, 1173
custom format
data types, displaying, 125-139
placeholders, 125-127
custom libraries, converting (Access 2.0), 1157-1159

Oracle Ex. 1009, p. 1251
Oracle v. DeBellis
IPR2024-00221

Customer.xls
Excel 97, creating, 1102-1103
named range of cells, specifying, 1111
worksheet objects, opening, 1109-1111
customizable toolbars, 447-450
creating, 450
Customize dialog box, 51, 77
Customize Toolbars dialog box, 448
customizing
Access 97 toolbars, 75-78
graphs, 724-727
Cut command (Edit menu), 72, 165, 170, 407, 433, 524

D

Dartmouth College, BASIC development, 978
data
DDE (dynamic data exchange), 1174
definitions, 1173
dictionaries, 1173
elements, 1173
intersections, 1183
entities, 1173
objects, 791-792
integrity, 1173
modeling databases, 797-802
modification, 1173
output field properties windows, formatting, 347-349
queries, transferring, 276-278
sharing, 1173
tables
editing, 53-56
printing, 59-62
selecting, 53-56
viewing, 52-53
Word 97 automation server, sending, 1116-1123
Data Access dialog box, 38, 922
Data Access Objects, 681, 1173
variable tags, 1222-1224
creating references, 1025-1027
hierarchy, 1029-1030
QueryDef objects, 1029
Recordset objects, 1030
relation objects, 1030
TableDef objects, 1029
data dictionaries, 818-823
data entry
expressions, validating, 310
keyboard, 162-168
configuring, 167-168
key combinations, 163-165
validation rules, 171-175
field-level validation rules, 171
table-level validation rules, 173-175
testing, 177-178
data event handlers source code, 1092-1093
data events, 1062-1068
responding (Access VBA), 1090-1094
data malls, 626-627
data marts
fixed disk drives
SCSI-2, 626-627
SCSI-3, 626-627
Microsoft Internet strategy, 626-627
Data menu commands, Series in Rows, 723
data modeling tools, creating, 800-802
data objects, 873
ActiveX, 1164
data pages, 825-826
indexes, 825-826
data sources
files, creating, 931-935
SQL Server
adding, 924-935
creating, 924-931
removing, 924-935
users, creating, 929-931
data type conversion functions, 294-295, 304
data types, 1174
display format
custom, 125-139
null values, 124-125
standard, 124
logical, 1186
placeholders, custom format, 125-127
selecting display format, 123-128
strings, 1200
tables, design grids, 116-117
user-defined data types, 1204
Database command (Window menu), 206
Database Documentor, 820-823
database maintenance forms, 396
database object data types (Access VBA), 986-988
database objects, 671
LNC, 1209
writing functions, 1041-1044
Database Splitter, 30, 874
Database toolbar, 52
Database Utilities command (Tools menu), 66, 102, 162, 1155
Database window, 1174
databases, 793-796, 1174
Access 97 format, converting, 104
Access 97 utilities, 102-105
Access master file, System.mdw, 112
administrating, 910-912
administrators, 1174
application characteristics (Access 97), 109-110
application databases, 798
application categories, 396
attribute data types, 795
attribute domains, 795
attribute values, 795
attributes, 794
backing up, 910-912
client/server databases, connecting to, 915

Oracle Ex. 1009, p. 1252
Oracle v. DeBellis
IPR2024-00221

environment, 915
exporting tables, 945-953
ODBC (Open Database Connectivity), 917-918
compacting, 102-104, 911
converting
Access 95, 1159-1160
creating, 112-114
data modeling tools, 800-802
queries, 262-279
current databases, 1173
custom format for data types, 125-139
data
grouping, 551-555
sorting, 551-555
data modeling, 797-802
data description, 794
data storage, 794
decryption, 911-912
default databases, 1175
designing, 788-789
object-oriented design, 790-793
devices, 1174
display format
data types, 123-128
null values, 124-125
distributed databases, 1176
encryption, 911-912
entities, 794
relationships, 809-818
field size property, selecting, 120-123
fields, 163
file formats, identifying, 205
grouping data
by category, 552-553
by range, 554-572
heterogeneous universe, 795
homogeneous universe, 795
identifiers, 795
integrity
enforcing, 829-834
referential integrity, 834
Leszynski Naming Conventions, 1208-1209
library databases, 1185
master databases, 1186
.mdb extension (Access 97), 111
.mde files, creating, 105
mission-critical, 1187
normalization, 1189
Northwind.mdb database, copying, 162-163
objects, 1174
naming conventions, 1215-1219
ODBC (Open Database Connectivity), 914, 935-940, 1190
OLE DB, 1191
opening, 48-50
Access 97, 50-51
owners, 1174
ownership, 892-895
changing, 892-897
passwords, 883-884
permissions, 892
granting, 895-902
workgroups, 902-910
RDBMS, 203
referential integrity, 151-152
relational, 110-111
relational databases, 787
designing, 790
keys, 796-797
normalization, 802-813
objects, 787-788
tables, 796-797
repairing, 104-105, 911
replicating, 971-973
restoring, 910-912
setting fixed-width text fields, 122
shared databases, opening, 877-881
sharing networks, 902-910
splitting, file sharing, 873-875
subject databases, 798
system databases, 1201
systems, 788-800
tables
creating in other, 861-862
linking, 217-218
relationships, 146-147
text data types, 795
three-tier architecture, 1202
window objects
level one tags, 1215-1217
level two tags, 1216-1217
windows (Access 97), 51-52
Datasheet command (View menu), 53, 716
Datasheet Formatting toolbar, 74-75
Datasheet view
button, 831
creating tables, 134-137
customizing, 199-201
options, setting, 88-89
Datasheet View command (View menu), 72, 146
Date and Time command (Insert menu), 765
Date and Time dialog box, 765
date function, 1174
date/time functions, 294-295, 300-301
date/time literals, 293
DATETIME reserved word (Jet SQL), 841
dBASE
comparisons with Access VBA, 979-980
files, identifying, 205
index files, linking, 213
DBEngine, 681
dbsamp2.htm source code, 632-633
dbWeb 1.1a
downloading, 579
features, 579
Microsoft Internet products, 579
DCL (Data Control Language), 839
DCOM (Distributed COM), 670, 1171
COM terminology, 1097
DDE (dynamic data exchange), 667, 1174
developmental history, 1095-1096
linking, advanced options, 86-88
OLE, compared, 675-676

Oracle Ex. 1009, p. 1253
Oracle v. DeBellis
IPR2024-00221

DDL (Data Definition Language), 801
SQL, 838
deadlocks, 1175
Debug menu commands
Add Watch, 1011
Clear All Breakpoints, 1011
Compile All Modules, 1043
Compile Loaded Modules, 1009
Edit Watch, 1011
Run to Cursor, 1011
Show Next Statement, 1011
Step Into, 1009
Step Over, 1009
Toggle Breakpoint, 1009
Debug window, 26, 1175
automated printing, 1018-1019
compilation results, displaying, 1016-1017
functions, 295-298
implementing object variables, 1033-1034
Debug Window command (View menu), 295, 1009
debugging, 1175
function syntax, 295-298
text manipulation functions, 303-304
decimal places, tables, field properties, 117-118
decision support forms, 396-397
declaration sections, 1175
declarations, 1175
DECLARE reserved word (ANSI SQL), 847
declaring
Access VBA arrays, 992-993
text, 1175
VBA object names, 1227
default databases, 1175
default ODBC data source, specifying, 642-643
Default Open Mode, multiuser applications, 872
default options
Datasheet view, setting, 88-89
setting in Access 97, 81-89
Default Record Locking, multiuser applications, 872
default values
expressions, creating, 309-310
setting in fields, 145-146
defaults, 1175
defining
database elements, 107-110
functions overview, 63-66
Delete key, 164
Delete Query command (Query menu), 382
Delete Record command (Edit menu), 74, 434
DeleteObject method, 1071-1075
deleting
AutoFormats, 412-413
buttons on toolbars, 449
controls
forms, 424
reports, 526-532
data sources (SQL Server), 924-935
fields
forms, 439
redundant, 253
groups, 891-892
records
Form view, 437
tables, 170, 381-383
reports
footers, 557-559
headers, 557-559
Tab control pages, 492
tables, 160
users, 891-892
delimited files, 242-243
demand locks, 1175
Denali, *see* ActiveX, server framework, 1164
dependents in tables, 1175
design grids
data type in tables, 116-117
field name in tables, 116-117
primary keys in tables, 116-117
setting in tables, 116-117
Design mode, 66-67, 1175
tables, implementing, 57-59
design properties, modifying (graphs), 715-720
Design View button, 172
Design View command (View menu), 72, 146, 433
design-master replicas, 1175
designing
databases, 788-789
object-oriented design, 790-793
Personnel Actions Table, 138-141
query parameters, 358-361
Desktop Database Drivers, 38
destination documents (OLE), 1175
detail level, modifying for crosstab queries, 365-367
detail tables, 1176
Details command (View menu), 115, 969
device context, 1176
DHCP (Dynamic Host Configuration Protocol), 1176
diagrammatic data models, 799
Dial-Up Networking Server, 958
dialog boxes, 1176
About, 91
Access 97 Options, 646
Add ActiveX Control, 1123
Add Reference, 1157
Add Tables, 263
Applicable Filter, 198
Cells Effects, 200
Change Access Rights, 905
Change Folder, 34
Chart Type, 717
check boxes, 1169
Checking and Reporting Errors, 776
Choose Columns, 773
Choose Data Source, 771
Choose Profile, 571
Command Button Wizard, 1057
common, 1171
Compact Into, 162
Component Builder, 1142
Confirmation, 178
controls, 1172
Copy Progress, 40
Create Data Source, 771

Oracle Ex. 1009, p. 1254
Oracle v. DeBellis
IPR2024-00221

Create New Data Source, 639, 930
Customize, 51, 77
Customize Toolbars, 448
Data Access, 38, 922
Database Splitter, 874
Date and Time, 765
Edit Hyperlink, 586
Enter Parameter Value, 332, 359
Export, 638
File New Database, 113
Filter Data, 774
Find, 80, 81
Find in Field, 183, 185
Font, 199
Format Axis, 717
Format Chart Title, 716
Format Data Series, 724
Format Legend, 716
Groups, 891
Help Topics, 36, 91-96
Import, 732
Import HTML Wizard, 620
Import/Export Specification, 620
Index, 209
Input Mask Wizard, 129
Insert ActiveX Controls, 1125
Insert Object, 687, 703
Insert Picture, 416
Join Properties, 157, 341
Link, 208
Link Tables, 217, 936
Link to File, 588
Linked Table Manager, 874
Logon, 886
Mail Merge Helper, 771, 777
Maintenance, 41
Make Table, 280, 374, 737
Missing Memo File, 210
model dialog boxes, 1187
modeless dialog boxes, 1187
Name and Organization, 32
New, 113, 959
New Database, 732
New Form, 399
New Query, 312, 747
New Report, 516, 568
New Style Name, 413
New Table, 130
Object Browser, 81
ODBC Create Data Source Administrator, 928
ODBC Microsoft Access 97 Setup, 773
ODBC Microsoft Access Setup, 640
ODBC SQL Server Setup, 925
Open, 49-50, 169
Options, 82
Page Setup, 60, 275, 544
Password Required, 208, 883
Paste As, 702
Paste Special, 678
Paste Table As, 160, 333, 746
Permissions, 902
Print, 546
Printer Setup, 514
Progress Monitor, 40
Query Parameters, 361
Relationships, 157, 376
Replace, 80-81
Replace in Field, 186
Report Wizard, 518
Run, 32
Save As, 135, 151, 255
Save As Query, 197
Select Data Source, 638, 943
Select a Jet Database, 1148
Select Microsoft Word Document, 768
Select Workgroup Information File, 876
Send, 570
Set Database Password, 883
Setup, 33
Show Table, 156, 193, 263, 440, 744, 842
Sort Order, 774
Sorting and Grouping, 546, 551
SQL Server Login, 934, 943
Startup, 31
Suppliers Import Specification, 620
Table Wizard, 131
Table Wizard opening, 131
Table Wizard's relationships, 132
Unset Database Password, 883
Update My Briefcase, 970
User and Group Permissions, 901
Workgroup Administrator, 871, 876-877
Workgroup Owner Information, 870

.dib (device independent bitmap) files, 699, 1176
dictionaries, data, 818-823
differing data sources, creating merge documents, 767-770
direct invocation (SQL), 837
directory lists, 1176
DirectX, 582
disks
mirroring, 1176
striping, 1176
display format
data types, selecting, 123-128
null values, data types, 124-125
displaying
custom format, data types, 125-139
Debug window compilation results, 1016-1017
documents in forms, 781-784
object models (Word 97), 1116-1118
Paint 95 graphics, 56-57
Personnel Actions, employee pictures, 693-696
query sorts, 267-269
standard format, data types, 124
DISTINCT reserved word (Jet SQL), 845
DISTINCTROW reserved word (Jet SQL), 841, 845
DISTINCTROW SQL reserved word (Jet SQL), 842
Distributed COM (DCOM)
Microsoft Internet technologies, 581
working specifications, obtaining, 581

Oracle Ex. 1009, p. 1255
Oracle v. DeBellis
IPR2024-00221

distributed databases, 1176
Distributed Transaction Controller (DTC), 627
distributed transactions
managing, 627
Microsoft Internet strategy, 626-627
Microsoft Transaction Server (MTx), 627
distribution disks
images, creating, 1138-1150
Setup Wizard, creating, 1138-1147
division (/) operator, 290
DLLs (dynamic link libraries), 1177, 1191
DLookUp( ) domain aggregate function, printing, 532-534
DML (Data Manipulation Language), 860-861
SQL, 838
Do Until...Loop, 1003
Do While...Loop, 1003
docfiles, 1177
DoCmd methods
AddMenu, 1070-1075
ApplyFilter, 1071-1075
arguments, 1070-1075
Beep, 1071-1075
CancelEvent, 1071-1075
Close, 1071-1075
DeleteObject, 1071-1075
Echo, 1071-1075
FindNext, 1071-1075
GoToControl, 1071-1075
GoToRecord, 1072-1075
Hourglass, 1072-1075
implementing in Access 97, 1068-1070
MoveSize, 1072-1075
OpenForm, 1072-1075
OpenModule, 1072-1075
OpenQuery, 1072-1075
OpenReport, 1072-1075
OutputTo, 1072-1075
Rename, 1073-1075
RepaintObject, 1073-1075
Requery, 1073-1075
RunCode, 1073-1075
RunCommand, 1073-1075
RunMacro, 1073-1075
RunSQL, 1074-1075
SendKeys, 1074-1075
SendObject, 1074-1075
SetMenuItem, 1074-1075
SetValue, 1074-1075
SetWarnings, 1074-1075
ShowAllRecords, 1074-1075
StopAllMacros, 1074-1075
TransferDatabase, 1075
TransferSpreadsheet, 1075
Document Control-menu box, 1177
Document Object command (Edit menu), 783
document windows (Access 97)
maximizing, 69-70
minimizing, 70-71
documents, 1177
ActiveX, 1164
creating in forms, 781-784
embedding in tables, 779-781
linking in tables, 779-781
Documents command (Start menu), 668
domain aggregate functions, 308-309, 1177
DLookUp( ), 532-534
domains, 1177
integrity, 1177
Don't Move keyboard option, 168-170
double precision, Excel 97 numeric fields, 734-735
DOUBLE reserved word (Jet SQL), 841
downloading
dbWeb 1.1a, 579
FrontPage 97 beta versions, 579-580
Index Server 1.1, 578
Internet Explorer, 576
NetMeeting 2.0, 576-577
ProxyServer, 577-578
DQL (Data Query Language), 838
drag-and-drop (mouse), 1177
drivers, 38
ODBC, 918-920
drives, 1177
DROP VIEW reserved word (ANSI SQL), 847
DTCs (Distributed Transaction Controllers), 917
dynamic databases, properties, 791
dynamic arrays, creating (Access VBA), 992
dynamic data exchange functions, 308-309
dynamic Web pages, creating, 643-651
dynasets, 1178

E

EBCDIC (Extended Binary-Coded-Decimal Interchange Code), 233
Echo method, 1071-1075
ECP (Extended Communication Protocol), 957
Edit Hyperlink command (Hyperlink menu), 586
Edit Hyperlink dialog box, 586
Edit menu commands
Copy, 73, 165, 407, 433, 524
Cut, 72, 165, 170, 407, 433, 524
Delete Record, 74, 434
Document Object, 783
Find, 84, 183, 433
Go To, 434
Paste, 73, 160, 165, 170, 407, 433, 524
Paste Append, 170
Paste Special, 678
Redo, 1009
Rename, 279, 940
Replace, 84, 186
Select All, 540
Select All Records, 276, 379
Select Form, 420, 423, 428
Select Report, 536, 559
Undo, 73, 160, 407, 433, 524, 1009
Worksheet Object, 758

Oracle Ex. 1009, p. 1256
Oracle v. DeBellis
IPR2024-00221

Edit Watch command (Debug menu), 1011
edit/find options, system default options, 84-85
editing
Categories table, graphics, 56-57
controls in reports, 526-532
data in tables, 53-56
fields
keys, 163
tables, 271
form elements, 419-426
grids, keys, 163
hyperlinks, 586-588
records, 436
text boxes, keys, 163
editing keys, 163-165
embedded objects, 1178
embedded SQL, 837
embedding
Excel 97 worksheets, 755-758
source documents, 676-679
tables in documents, 779-781
employee pictures (Personnel Actions)
displaying, 693-696
forms, troubleshooting, 695-696
empty toolbars, creating, 449
empty variables, 1178
emulating
QueryDef object, Recordset object, 1039-1040
replica sets, reconciliation, 967-971
enabled objects, 1178
encryption databases, 911-912
End command (Run menu), 1009
enforcing referential integrity in tables, 151-152
Enter Parameter Value dialog box, 332, 359
entering data via keyboard, 162-168
entities
databases, 794
integrity, enforcing, 829-834
relationships, 809-818
environmental variables, 1178
environments, 1178
event-driven, 1179
equi-joins, 323, 1178
single column, creating, 323-326
Err function, error detection (Access VBA), 1006-1007
Error event
form modules, implementing, 1007
report modules, implementing, 1007
error events, 1062-1068
error trapping, 1178
Esc key, 164-165
establishing
cascading deletions, 388-389
cascading updates, 388-389
table relationships, 146-147, 376-377, 750-753
event handling capabilities, Access prior versions, 1049
event handling codes
adding, 1056
macro conversion process, 1083-1086
main switchboard source code, 1053-1054
viewing, 1050-1054
event-driven
environments, 1179
operating systems, 1179
events, 1178
categories
class module, 1062-1068
data, 1062-1068
error, 1062-1068
filter, 1062-1068
focus, 1062-1068
keyboard, 1062-1068
mouse, 1062-1068
print, 1062-1068
timing, 1062-1068
window, 1062-1068
functions, responding, 1060-1061
handling, 1049
example worksheet (Excel 97)
importing, 732-735
obtaining, 731-732
re-creating, 746-750
Excel
programmable objects (VBA hierarchy), 1103-1104
spreadsheets
importing, 220-233
linking, 220-233
worksheets, linking, 228-233
Excel 97
cell ranges, linking, 760-762
creating Customer.xls, 1102-1103
creating worksheets (Access VBA), 1112-1115
example worksheet, obtaining, 731-732
implementing as OLE servers, 754-762
numeric fields
double precision, 734-735
single precision, 734-735
Online Help File, launching, 1104-1106
VBA, 1104-1106
opening worksheet objects (Customer.xls), 1109-1111
programmable objects
manipulating, 1102-1112
referencing, 1106-1108
records, exporting, 754
spreadsheet data conversion in tables, 735-737
worksheet data, importing, 730-731
worksheet objects, closing, 1111-1112
worksheets, embedding, 755-758
exclusive locks, 1179
executable code, 1179
executable files, 28
executing
Access 2.0 applications in Access 97, 1152-1154
Access VBA functions, 984-985
calculations
selected table records, 356-359
table records, 352-356
tables, test query, 752-753

Oracle Ex. 1009, p. 1257
Oracle v. DeBellis
IPR2024-00221

existing databases, adding to tables, 137-138
Exit & Return command (File menu), 705
Exit command (File menu), 56, 676
explicit permissions, 896
explicit variables (Access VBA), 989
explicitly closing Excel 97 worksheet objects, 1111-1112
exponents, 1179
Export command (File menu), 945
Export dialog box, 638
Export to HTML command (File menu), 611
exporting
  Excel 97 records, 754
  forms (ASP), 660-662
  queries
    ASP, 655-660
    HTML, 597-600
  records
    Clipboard, 253-257
    filtered, 201-202
    sorted, 201-202
    tables, 253-257
    as text files, 255-257
  reports (HTML), 610-613
  tables, 945-946
    client/server databases, 945-953
    HTML, 597-600
  templates in queries, 608-610
Expression Builder, 447
  adding query criteria, 314-316
expressions, 1179
  creating, 171, 309-319
    default values, 309-310
  defined, 288-289
  elements
    functions, 288-289
    identifiers, 288-289
    literals, 288-289
    operators, 288-289
  operators, 171
    arithmetic, 171
    comparison, 171
    logical, 171
  query criteria
    creating, 312-314
    setting, 310-316
  query field values, calculating, 316-319
  sorting data, 555
  troubleshooting, 319
  validating data entry, 310
external data sources
  querying, 367-369
  verifying in tables, 746-750
external tables
  graphics, 210-212
  importing, 206-209
  linking, 206-209
extracting values from OLE objects, 759-760

F

F2 key, 163-164
F8 key, 165
failover, 1179
FAT (File Allocation Table), 1179
fault tolerance, 1179
FETCH reserved word (ANSI SQL), 847
fibers, 1179
field base names, objects, 1214
field data types
  Access VBA, 986-988
  hyperlinks, 585
  listings, 119-120
  modifying in tables, 156-158
  numeric, modifying, 157-158
  tables, selecting, 119-120
  text, modifying, 158
field information, assigning (Personnel Actions Table), 140-151
Field List command (View menu), 407, 524
field properties (tables)
  allow zero length, 118-119
  caption, 117-118
  decimal places, 117-118
  format, 117-118
  indexed, 118-119
  input mask, 117-118
  setting in Personnel Actions Table, 141-144
  size, 117-118
  validation rules, 118-119
  validation text, 118-119
Field Properties window, 172
field size property
  selecting
    numeric data, 120-123
    text data, 120-123
field-level validation rules, 171
fields, 163, 1180
  altering in tables, 155
  asterisk, adding, 350
  data types, changing, 252
  default values, setting, 145-146
  editing keys, 163
  field list title bar, including, 349-350
  forms
    deleting, 439
    ordering, 438-442
  freezing in tables, 180-181
  lookup fields, 30
  OLE Object data types, 122-123
  output all fields property, selecting, 350-351
  properties, overriding, 509-510
  queries, selecting, 264-267
  rearranging sequences, 155-156
  records, sorting, 181-182
  redundant deleting, 253
  selecting reports, 517
  tables
    accessing, 349-351
    editing, 271
  text, selecting, 164-165
  thawing tables, 182-183
  validation rules, 171-173
  Yes/No fields, 1206
fields function keys
  assigning, 80
fields, grids and text boxes, Access 97, function keys, 80
fifth normal form rule, 807-808, 1180

Oracle Ex. 1009, p. 1258
Oracle v. DeBellis
IPR2024-00221

File Data Source page, ODBC Administrator tool, 929
file formats
bitmap images, 699-701
query results, printing, 273-276
File menu commands
Close, 51, 162
Exit, 56, 676
Exit & Return, 705
Export, 945
Export to HTML, 611
Get External Data, 207, 217, 618, 732, 892, 1156
Import, 618
Load Database File, 1148
Load from Query, 198
New, 113, 160
New Database, 207, 732
Open, 163
Open Database, 50, 206
Page Setup, 60, 275, 544, 783
Print, 60, 72, 433, 544
Print Preview, 59, 72, 407, 433, 523
Properties, 668
Return Data to Microsoft Word, 776
Save, 72, 79, 407, 433, 523
Save As, 279
Save As HTML, 614, 643
Save As Query, 197
Save As/Export, 597, 655
Save Copy As, 705
Send, 570
Undo, 54
file monikers, 1180
File New Database dialog box, 113
file paths, hyperlinks, 584
file sharing, 868-877
accessing, 907-910
administrating, 910-912
databases, splitting, 873-875
folders, creating, 868-872
groups
deleting, 891-892
managing, 884-892
networks, 902-910
servers, 907
Windows 95, 902-904
ownership, 892-895
changing, 892-897
preparations, 872-873
security, 881-892
LANs, 881
passwords, 883-884
shared databases, opening, 877-881
system files, creating, 868-872
user-level security, 904-906
users
deleting, 891-892
managing, 884-892
files, 29, 1180
backups, 1166
batches, 1167
binary files, 1167
CDFS (CD-ROM file system), 1169
compacting, 911
data sources, creating, 931-935
dBASE, identifying, 205
delimited files, 242-243
docfiles, 1177
encryption, 911-912
executable files, 28
exporting, 253-257
FAT (file allocation table), 1179
formats
handling, 204-220
identifying, 205
FoxPro 2+, identifying, 206
header files, 1182
HPFS (High-Performance File System), 1182
importing
spreadsheets, 220-233
text files, 233-245
troubleshooting, 209-210
index files, 213-217
linking
spreadsheets, 220-233
troubleshooting, 209-210
links, updating, 679
.mdb, 866
Jet 3.5, 28
.mde, 26
metafiles, 1187
NFS (Network File Server), 1189
NTFS (New Technology File System), 1189
Paradox, identifying, 206
Registry, replacing with backups, 687-689
repairing, 911
RIFF (Resource Interchange File Format), 1196
sequential access file, 1198
spreadsheets
importing, 220-233
linking, 220-233
support files, 28
text files, 1202
fixed-width text files, 244
importing, 233-245
WAVE files, 1205
Filter by Form, 30
Filter by Form button, 190
Filter by Selection, 30
Filter command (Records menu), 73, 187, 433
Filter Data dialog box, 774
filter events, 1062-1068
Filter window, 197
filtering records
advanced filters, 193
composite filters, 195-197
compound filters, 193-195
copying, 201-202
exporting, 201-202
by form, 189-192
loading, 197-202
mailing, 201-202
saving as queries, 197-202
by selection, 187-189
tables, 186-199
filters, table properties, 115
financial functions, 294-295, 306-307
Find command (Edit menu), 84, 183, 433
Find command (Start menu), 940
Find dialog box, 80-81
Find First button, 184
Find in Field dialog box, 183, 185
Find tab (Help Topics dialog box), 96
Find What text box, 183

Oracle Ex. 1009, p. 1259
Oracle v. DeBellis
IPR2024-00221

finding record matches, 183-186
FindNext method, 1071-1075
first normal form rule, normalization, 803
First Record button, 187
fixed column headings, implementing crosstab queries, 363-365
fixed disk drives (data marts)
  SCSI-2 drives, 626-627
  SCSI-3 drives, 626-627
fixed value lookup lists, adding, 337-340
fixed-width text fields, setting, 122
fixed-width text files, 244
flags, 1180
flow control, 1180
focus, 1180
focus events, 1062-1068
folders, creating file sharing, 868-872
Font command (Format menu), 75, 199
Font dialog box, 199
fonts, 1180
footers
  adding
    forms, 510
    reports, 557-559
  deleting in reports, 557-559
  forms, sizing, 421
  printing in reports, 559
For...Next statements, 1002-1003
  array values, assigning, 1003
foreground colors
  forms, changing, 417-421
foregrounds, multitasking, 1180
foreign-key drop-down list
  adding lookup fields, 333-337
foreign-key values, table lookup fields, 332-333
foreign keys, 1180
  relational databases, 797
Form command (View menu), 407
Form Design toolbar, 405
Form Design view, controls, appearance, 450-451
Form Design window, 404-419
  commands, 406-408
  default values, 410
  elements, 405-406
  Formatting toolbar, 409-410
  toolbar, 406-408
Form Detail bar (Form Design window), 406
Form Footer bar (Form Design window), 406
Form Header bar (Form Design window), 406
Form Header/Footer command (View menu), 406, 454
form letters (Mail Merge Wizard)
  creating, 764-767
  previewing, 764-767
  queries, creating, 777-778
form level scope for Access VBA variables, 990-991
form modules
  Access VBA, 981-985
  implementing Error event, 1007
Form objects, 671
Form view
  records, deleting, 437
  toolbar, 433
Form Wizard, 447
  forms, creating, 399-404
  transaction processing forms, creating, 397-404
form-level variables, 1180
Format Axis dialog box, 717
Format Chart Title dialog box, 716
Format Data Series dialog box, 724
Format Legend dialog box, 716
Format menu commands
  AutoFormat, 407, 524
  Cells, 75, 200
  Change To, 495
  Font, 75, 199
  Freeze Columns, 180
  Hide Columns, 199
  Row Height, 200
  Selected Axis, 717
  Selected Chart Title, 716
  Size, 422
  Unfreeze All Columns, 183
  Unhide Columns, 199
Format Painter, 426
formatting
  controls for reports, 540
  output field properties window, 347-349
  text in forms, 457-458
Formatting toolbar (Form Design window), 405, 409-410
FORML (Formal Object Role Modeling Language), 801
forms, 395, 1180
  adding
    bound object controls, 692-701
    Calendar Control, 1124-1127
    unbound object frames, 701-707
  ASP, exporting, 660-662
  AutoFormats, 411-413
  blank, creating, 454-455
  borders, changing, 417-421
  check boxes, 464-472
  colors
    background colors, 414
    changing, 413-421
    foreground colors, 417-421
  combo boxes, 474-486
    adding, 474
  completing, 495-499
  configuring, 427-429
  continuous forms, creating, 503-509
  controls
    adding, 444-446, 451-495
    aligning, 424
    aligning in grids, 422-423
    binding, 464-472
    categories, 444
    changing borders, 424-425
    changing types, 495
    copying, 463

Oracle Ex. 1009, p. 1260
Oracle v. DeBellis
IPR2024-00221

copying to other forms, 472-474
deleting, 424
moving, 421-424
selecting, 421-424
sizing, 421-422
Tab controls, 489-495
creating (Form Wizard), 399-404
decision support forms, 397
documents, creating, 781-784
elements
editing, 419-426
moving, 419-426
selecting, 419-426
fields
deleting, 439
ordering, 438-442
filtering records, 189-192
footers
adding, 510
sizing, 421
Format Painter, 426
headers
adding, 510
labeling, 455-457
sizing, 421
list boxes, 474-486
modifying, 437-442
option buttons, adding, 464-472
Personnel Actions form, 433
queries, creating, 452-454
rearranging, 426-431
reports, compared, 513-514
selecting, 398
SQL statements, 863-864
subforms, 1200
configuring, 430-431
creating, 499-502
text, formatting, 457-458
text boxes, creating, 458-463
text controls, editing, 425-426
toggle buttons, 464-472
tools, locking, 464-472
transaction processing forms, 396-397, 433-437
creating, 397-404
troubleshooting employee pictures, 695-696
types, 396-397
views, changing default, 464
fourth normal form rule, 806-807, 1181
FoxPro 2+ files, identifying, 206
frames, 1181
objects, 1190
Freeze Columns command (Format menu), 180
freezing fields in tables, 180-181
front-end, 1181
FrontPage 97
beta versions, downloading, 579-580
features, 579-580
Microsoft Internet products, 579-580
full permissions, 896
full servers, 1181
full-server applications, 669
function keys (Access 97), 79-81
global, 79
module window, 80-81
functions, 1164, 1181
Access VBA, 984-985
aggregate functions, 1177
built-in functions, 1168
closed queries Access 97, 66
data type conversion, 294-295, 304
database objects, writing, 1041-1044
date, 1174
date/time, 294-295, 300-301
Debug window, 295-298
defined, 294-295
domain aggregate, 308-309
domain aggregate functions, DLookUp( ), 532-534
dynamic data exchange, 308-309
executing (Access VBA), 984-985
expression element, 288-289
financial, 294-295, 306-307
general purpose, 294-295
IsNull(), 841
Jet SQL, 840-842
libraries, 1185
macros (Access 97), 65-66
mathematical, 294-295, 304-306
miscellaneous, 307-308
modules (Access 97), 65-66
nilTestQuery, 1041-1044
ODBC Databases( ), 945
OLE, 956-957
open queries, 66
printing, 66
responding events, 1060-1061
security, 66
SQL aggregate, 308-309
SQL aggregate functions, 852-855
SQLExecDirect( ), 944
StdDev( ), 842
StdDevP( ), 842
syntax, debugging, 295-298
system functions, 1201
text manipulation, 294-295, 302-304
trigonometric, 294-295, 304-306
Var( ), 842

G

Gates, Bill
Access VBA, developmental history, 978-979
Microsoft strategy for Internet, 575-576
general purpose functions, 294-295
Get External Data command (File menu), 207, 217, 618, 732, 892, 1156
.gif files, 700
global function keys, assigning, 79
global level scope for Access VBA variables, 990-991
global modules, 1181

Oracle Ex. 1009, p. 1261
Oracle v. DeBellis
IPR2024-00221

global variables, 1181
Go To command (Edit menu), 434
Go to End of Field keyboard option, 168
Go To New command (Records menu), 74
Go to Start of Field keyboard option, 168
Go To statements
BASIC, 998
"spaghetti code," 999
GoToControl method, 1071-1075
GoToRecord method, 1072-1075
GRANT reserved word (ANSI SQL), 847
granting permissions, 895-902
Graph 8.0, 25
graphic images
including (Personnel Actions), 692-693
scaling (Access 97), 696-699
graphical QBE expressions, translating, 284-286
graphics
bitmaps, 1167
displaying in Paint 95, 56-57
editing in Categories table, 56-57
external tables, 210-212
icons, 1182
persistent graphics, 1192
viewing in Categories table, 56-57
graphics images
adding bound object frames, 698-699
clipping, 698-699
zooming, 698-699
graphics interchange format, *see* **.gif files graphs (Microsoft Graph 97)**
creating, 709-727
crosstab queries, creating, 721-723
customizing, 724-727
design features, modifying, 715-720
line-item data, 709-711
summary data, 709-711
tables, linking, 723-727
greater than (>) operator, 291
greater than or equal to (>=) operator, 291
Grid command (View menu), 423
grids, 163, 1181
editing keys, 163
form controls, aligning, 422-423
text, selecting, 164-165
grids function keys, assigning, 80
group aggregate functions (SQL), 839
GROUP keyword (SQL), 853
grouping
data
by category, 552-553
by range, 554-572
LNC objects, 1209-1210
reports, 516-523
groups, 1181
adding, 890-891
administrators, 884
deleting, 891-892
managing, 884-892
members, establishing, 889-890
permissions
changing, 897-899
granting, 895-902
revoking, 895-902
Groups dialog box, 891
groups/totals reports, 515
guests, 884

H

HALs (Hardware Abstraction Layers), 1181
handles, 1181
sizing handles, 1199
handling
Access VBA, run-time errors, 1005-1007
events, 1049
file formats, 204-220
hardware, monitors, 1188
HAVING keyword (SQL), 853
headers, 1182
adding
forms, 510
reports, 557-559
deleting in reports, 557-559
forms
labeling, 455-457
sizing, 421
printing in reports, 559
Help menu options, 91
Help menu commands
Access, 91
Annotate, 97
Contents and Index, 91, 297
Keep Help, 97
Microsoft Access, 408
Microsoft Access Help, 98, 434
Microsoft on the Web, 91
Print Topic, 97
Use System, 97
What's This?, 90
help system
Access VBA Help System, 1011-1012
context-sensitive help (Access 97), 90-91
Office Assistant, 97-102
overview, 89-102
Help Topics dialog box, 36, 91-95
Contents tab, 92
Find tab, 96
Index tab, 95-96
options, 91-97
Help windows
hotspots, 94-95
options, 97
repositioning, 92-93
sizing, 92-93
heterogeneous universe databases, 795
Hide Columns command (Format menu), 199
hierarchies
data access objects, 1029-1030
menus, 1182
objects, 672-673
High table, creating, 743-746
hint screens (Report Wizard), 518

Oracle Ex. 1009, p. 1262
Oracle v. DeBellis
IPR2024-00221

history (SQL), 837-838
HKEY_CLASSES_ROOT subtree (Registry Editor), 684
HKEY_CURRENT_CONFIGURATION subtree (Registry Editor), 684
HKEY_CURRENT_USER subtree (Registry Editor), 684
HKEY_DYN_DATA subtree (Registry Editor), 684
HKEY_LOCAL_MACHINE subtree (Registry Editor), 684
HKEY_USERS subtree (Registry Editor), 684
home key, 164
home pages, creating (Web Publishing Wizard), 616-618
homogeneous universe databases, 795
hosts, 1182
hotlinks, 1182
hotspots, 94-95
Hourglass method, 1072-1075
HPFS (High-Performance File System), 1182
HTML (Hypertext Markup Language), 1182
  Access 97 templates, viewing, 606-608
  ASP, 652
  .asp file source code, 658-660
  exporting
    queries, 597-600
    reports, 610-613
    tables, 597-600
  importing records, 618-623
  source codes, Suppliers (unformatted) file, 602-603
  templates (Access 97), 606-608
HTML Export, Web pages, creating, 643-651
.htx files, 628
  JobForum application, 635
Hughes Direct Broadcast System (DBS), 700
Hyperlink command (Insert menu), 120, 407, 433, 524
Hyperlink menu commands, Edit Hyperlink, 586
hyperlinks
  Access controls, implementing, 595-596
  Access objects, opening, 592-594
  development, 583-584
  editing, 586-588
  features, 583-584
  field data type, 585
  file paths, 584
  inserting, 586-588
  intranets, navigating, 584-585
  specifying
    intranets, 590-592
    URLs, 594-595
  tables, testing, 586

I

icons, 1182
  document windows, minimizing, 70-71
  drag-and-drop, 1177
  shortcuts, creating, 880
.idc files, 628
  default ODBC data source, specifying, 642-643
  JobForum application, 635
IDC (Internet Database Connectors)
  .htx files, 628
  .idc files, 628
  demonstration applications (Internet Information Server), 628-635
  dynamic Web pages, creating, 643-651
  parameterized query (dbsamp2.htm), 632-633
  sample .htx file, 634-662
  sample .idc files, 631-634
  sample2 .idc files, 631-634
  Web pages, creating, 627-637
identifiers, 1182
  databases, 795
  defined, 294
  expression element, 288-289
  name structure, 294
  operators, 292
idle states, 1182
If...Then...End If statements, 999
Image command (View menu), 696
image controls
  Access 95, 707-709
  Access 97, 707-709
  properties, modifying, 708-727
Image tool (Toolbox), 445
images
  bitmapped images, linking, 679
  creating distribution disks, 1138-1150
  including in Personnel Actions, 692-693
  scaling (Access 97), 696-699
  thumbnail images, 679
immediate windows, 1182
implementing
  Access 97
    automation servers, 1131-1133
    DoCmd methods, 1068-1070
    Object Browser, 1027-1028
  Access VBA, 980
  automation servers in Access 97, 1100-1102
  crosstab queries, fixed column headings, 363-365
  Design mode in tables, 57-59
  Error event
    form modules, 1007
    report modules, 1007
  hyperlinks, Access controls, 595-596
  input masks, 128-129
  object variables (Debug window), 1033-1034
  OLE servers in Excel 97, 754-762
implicit variables (Access VBA), 988-989
Import command (File menu), 618
Import Database Add-In, 894

Oracle Ex. 1009, p. 1263
Oracle v. DeBellis
IPR2024-00221

Import dialog box, 732
Import HTML Wizard dialog box, 620
Import Spreadsheet Wizard, 223
Import/Export Specification dialog box, 620
importing, 30
  example worksheet, 732-735
  Excel 97 worksheet data, 730-731
  field data types, changing, 252
  fields
    deleting redundant, 253
    modifying, 252-253
  linking, compared, 219-220
  records
    Clipboard, 247-250
    HTML tables, 618-623
  spreadsheets, 220-233
  tables
    external tables, 206-209
    modifying, 251-253
  text files, 233-245
    delimited files, 242-243
    fixed-width text files, 244
  troubleshooting, 209-210
IN clause (Jet SQL), 861-863
in-place activation (OLE), 680-681
in-process OLE, 1183
in-situ editing (OLE), 680
including
  fields in field list title bar, 349-350
  Personnel Actions, graphic images, 692-693
Increment AutoNumbers, 1166
Index dialog box, 209
index method object collections, 1030
Index Server 1.1
  downloading, 578
  Microsoft Internet products, 578
Index tab (Help Topics dialog box), 95-96
indexed tables, field properties, 118-119
indexes, 213-217, 823-829, 1183
  binary trees, 826-827
  clustered indexes, 827-828, 1170
  composite indexes, 1171
  data pages, 825-826
  ISAM (Indexed Sequential Access Method), 824-825
  multiple-field, creating, 154-155
  non-clustered indexes, 827-828, 1189
  per table limit, 1159
  querying, 828-830
  single-field, creating, 154-155
  tables, adding, 154-155
  unique indexes, 1204
  updates, 828-830
  variable length records, 825-826
Indexes command (View menu), 58, 215
indirect table relationships, creating queries, 326-329
\InetPub subfolders (ASP), creating, 652-655
infinite loops, 1183
  avoiding, 1004-1005
InfoModeler, 801
information criteria (Personnel Actions Table), 139-140
inheritance, objects, 672-673
initializing, 1183
inner queries, 1183
input mask, field properties, 117-118
Input Mask Wizard, 129
Input Mask Wizard dialog box, 129
input masks
  creating, 128-129
  implemeting, 128-129
  placeholders, 128-129
Insert ActiveX Controls dialog box, 1125
Insert Hyperlinks command (Insert menu), 73
Insert menu commands
  ActiveX Control, 1124
  Date and Time, 765
  Hyperlink, 120, 407, 433, 524
  Insert Hyperlinks, 73
  Module, 1008
  Name, 222, 1103
  Object, 703
  Procedure, 1021
Insert Object dialog box, 687, 703
Insert Picture dialog box, 416
inserting hyperlinks, 586-588
insertion points, 1183
installing
  Access to networks, 866-867
  Access 97, 32-42
  ODBC drivers, 920-923
  ODE (Office 97 Developer Edition) components, 44-45
  ORK (Office 97 Resource Kit), 44
  Partial Replica Wizard, 972
  SQL Server, 920-923
  ValuPack components, 42-43
instances, 1183
  objects, 1183
Instant Watch command (Tools menu), 1009
INT[EGER] reserved word (Jet SQL), 841
integers, 1183
integrating objects in relational databases, 787-788
integrity
  databases
    enforcing, 829-834
    referential integrity, 834
  domains, 1177
  referential, 1195
interacting
  Windows 95
    16-bit programmable objects, 1099-1100
    32-bit programmable objects, 1099-1100

Oracle Ex. 1009, p. 1264
Oracle v. DeBellis
IPR2024-00221

Windows NT
16-bit programmable objects, 1099-1100
32-bit programmable objects, 1099-1100
interfaces, 1183
CLI (Call Level Interface), 919
TDI (Transport Driver Interface), 1201
UI (user interface), 180
Internet
Access 97, 22
Active Server Pages, 582
ActiveMovie, 581
ActiveX, 581
ActiveX technology development, 1096-1097
Common Object Model (COM), 580
DirectX, 582
Distributed COM, 581
hyperlink development, 583-584
ISAPI, 582
Microsoft strategy, 575-576
data malls, 626-627
data marts, 626-627
distributed transactions, 626-627
OLE DB, 582
PC97, 583
Visual Basic, 582
see also Web; Web sites
Internet Database Connector, *see* **IDC**
Internet Explorer 3.0
cost, 576
downloading, 576
features, 576
Microsoft Internet products, 576-580
Internet Information Server (IIS)
features, 577
IDC demonstration applications, 628-635
Microsoft Internet products, 577
Internet Services Application Programming Interface, *see* **ISAPI**
interprocess communication (IPC)
DDE (dynamic data exchange), 666-667
developmental history, 1095-1096
intersections, data elements, 1183
intranets, 1184
hyperlinks, 584-585
specifying, 590-592
intrinsic constants, 309
Recordset objects, 1035
intrinsic symbolic constants, 996-997
inventory reports, creating, 516
invocation paths, 1183
invoking Access VBA procedures, 983
IP (Internet Protocol)
addresses, 1184
hosts, 1182
IPX/SPX (Internetwork Packet Exchange/Sequenced Packet Exchange), 914, 1184-1185
Is operator, 171
ISAM (Indexed Sequential Access Method), 824-825
ISAPI (Internet Services Application Programming Interfaces) Microsoft Internet technologies, 582
ISDN (Integrated Services Digital Network), 1184
IsNull( ) function, 841
items, 1184

J - K

Jet, ODBCDirect, 1190
Jet (Joint Engine Technology), 1184
Jet 3.5, 24
database engine, 1023
HTML tables, importing, 618-623
.mdb files, 28
Jet DDL, tables, creating, 846
Jet SQL
ANSI
compared, 838-847
non-supported keywords, 846-849
functions, 840-842
IN clause, 861-863
operators, 840-842
reserved words, 840-842
(date specifier), 842
<> (not equal), 842
? (LIKE wild card), 842
BINARY, 841
BOOLEAN, 841
BYTE, 841
CURRENCY, 841
DATETIME, 841
DISTINCT, 845
DISTINCTROW, 841, 845
DISTINCTROW SQL, 842
DOUBLE, 841
INT[EGER], 841
LONG, 841
LONGBINARY, 841
LONGTEXT, 841
OWNERACCESS, 841
PIVOT, 841
REAL, 841
SHORT, 841
TRANSFORM, 842
VARCHAR, 841
JobForum Web site, 635-637
Join Properties command (View menu), 341
Join Properties dialog box, 157, 341
joining multitable queries, 322-323
joins, 1184
creating with SQL, 855-857
equi-joins, 323, 1178
natural joins, 1188
not-equal joins, 1189
outer, 323
outer joins, 1191
self, 323
self-joins, 1198
Show Relationships command, 322-323
theta, 323
theta joins, 1202

Oracle Ex. 1009, p. 1265
Oracle v. DeBellis
IPR2024-00221

Joint Photographic Experts Group, *see* JPEG
Hughes Direct Broadcast System (DBS)
JPEG, Hughes Direct Broadcast System (DBS), 700
jumps, 1184

Keep Help command (Help menu), 97
Keep Together property, reports, 546-547
key combinations
Clipboard, 165-166
data entry keys, 163-165
editing
fields, 163
grids, 163
text boxes, 163
text, selecting, 164-165
keyboard
data entry, 162-168
configuring, 167-168
events, 1062-1068
options, setting system default options, 85-86
keys, 1184
composite keys, 1171
editing keys, 163-165
primary keys, 1193
setting, 213-217
relational databases, 796-797
values, 1184
keywords, 1184
ANSI non-support, 846-849
ANSI SQL, keyword substitutes, 840
operators, 1191
SQL, 839
ANSI, 840
Kodak Photo CD Access, 700

L

Label tool (Toolbox), 445
labels, 1185
form headers, adding, 455-457
LANs (local area networks), 1185
ProxyServer, 577-578
security, 881-882
laptops, 955
launching
Access 97, 48-50
applications, 1185
Excel 97 Online Help File, 1104-1106
Mail Merge Wizard, 764
Object Browser, 1027-1028, 1054-1056
Publish to the World Wide Web Wizard, 643
length limits
objects, base names, 1213
VBA objects, base names, 1219-1220
less than (<) operator, 291
less than or equal to (<=) operator, 291
Leszynski Naming Conventions, *see* LNC
level one tags
control objects, 1217-1218
window objects, 1215-1217
level two tags
control objects, 1217-1218
window objects, 1216-1217
libraries, 1185
objects, 1190
library databases, 1185
library references, storing (Access 97), 1158-1159
Like operator, 171
line graphs, converting, 717-720
line spacing, reports, adjusting, 541-543
Line tool (Toolbox), 446
line-item data, graphs, 709-711
Link dialog box, 208
Link Tables dialog box, 217, 936
Link to File dialog box, 588
linked graphs, defined, 711-715
linked objects, 1185
linked subreports, reports, adding, 564-568
Linked Table Manager, 874, 908
tables, relinking, 218-219
linked tables, 1185
linking
bitmapped images, 679
bookmarks (Word 97), 588-590
dBASE index files, 213
Excel 97 cell ranges, 760-762
graphs in tables, 723-727
importing, compared, 219-220
source documents, 676-679
spreadsheets, 220-233
tables, 217-218
documents, 779-781
external tables, 206-209
modifying, 251-253
troubleshooting, 209-210
worksheets (ODBC), 228-233
links
creating OLE, 760-762
hotlinks, 1182
troubleshooting, sub-reports, 568
List Box tool (Toolbox), 445
list boxes
combo boxes, compared, 474
forms, 474-486
listings
17.1 Suppliers (unformatted) file source code, 602-603
18.2 dbsamp2.htm source code, 632-633
18.4 sample.htx file source code, 634-635
18.6 HTML .asp file source code, 658-660
30.1 event-handling code for main switchboard class module, 1053-1054
30.2 event-handling code for macro conversion proc, 1083-1086
30.3 VBA validation subprocedure, 1091
30.4 data event handlers source code, 1092-1093
31.1 creating Excel workbook objects source code, 1113-1114
lists, 1185

Oracle Ex. 1009, p. 1266
Oracle v. DeBellis
IPR2024-00221

literal method, object collections, 1030
literals
date/time, 293
expression element, 288-289
numeric, 293
text, 293
livelocks, 1185
LNC (Leszynski Naming Conventions)
base names
creating, 1212-1215
length limits, 1213
naming rules, 1212-1213
complete text, obtaining, 1229
control objects, 1209
Data Access Objects, variable tags, 1222-1224
database objects, 1209
development, 994-995
field base names, objects, 1214
level one, 1209
tags, control objects, 1217-1218
level two, 1209
tags, control objects, 1217-1218
objects
base names, 1211-1212
compound base names, 1214
grouping, 1209-1210
naming conventions, 1215-1219
prefixes, 1210
suffixes, 1212
tags, 1211
procedure names
prefixes, 1226
tags, 1226
tags
components, 1228
creating, 1227-1228
VBA objects, 1209
Load Database File command (File menu), 1148
Load from Query command (File menu), 198
loading filters, 197-202
local level scope, Access VBA variables, 990-991
local variables, 1185
locking tools in forms, 464-472
locks, 1185
deadlocks, 1175
exclusive locks, 1179
livelocks, 1185
optimistic locking, 1191
pessimistic locking, 1192
shared locks, 1198
logging queries (ODBC), 944-945
logical data types, 1186
logical operators, 171, 291
logins, authentication, 1166
Logon dialog box, 886
logos, creating (Personnel Actions), 702-705
LONG reserved word (Jet SQL), 841
LONGBINARY reserved word (Jet SQL), 841
LONGTEXT reserved word (Jet SQL), 841
lookup fields, 30
foreign-key drop-down list, adding, 333-337
foreign-key values, 332-333
single primary key values, 332-333
tables, adding, 332-333
Lookup Wizard
adding fixed value lookup lists, 337-340
table look fields, creating, 333-337
looping operations
Do Until...Loop, 1003
Do While...Loop, 1003
For...Next statement, 1002
infinite loops, 1004-1005, 1183
termination, 1004
Low table, creating, 743-746
LRPCs (lightweight remote procedure calls), 1186

M

machine languages, 1186
Macmillan Publishing USA Information SuperLibrary Web site, 731
Macro command (Tools menu), 1082, 1131
macros, 1186
Access 97 functions, 65-66
prefixes, 1226-1227
SQL statements, 863-864
support, 28
tags, 1226-1227
VBA code, converting, 1080-1086
versus VBA, 1226-1227
versus VBA code, 977-978
mail merge data source, creating (Microsoft Query), 770-777
Mail Merge Helper dialog box, 771, 777
Mail Merge Wizard
features, 763-764
form letters
creating, 764-767
previewing, 764-767
launching, 764
mailing
labels, 515
multicolumn reports, printing as, 547-550
newspaper columns, 550
records
filtered, 201-202
sorted, 201-202
Maintenance dialog box, 41
Make MDE File command (Tools menu), 105
Make Table dialog box, 280, 374, 737
Make Table Query command (Query menu), 737, 743
make-table queries
building select queries, 372-373
converting select queries, 374-375
parameters, adding, 282-284
tables, creating, 371-378, 737-739

Oracle Ex. 1009, p. 1267
Oracle v. DeBellis
IPR2024-00221

managing distributed transactions, 627
manipulating
automation servers, Registry entries, 1108-1109
Excel 97 programmable objects, 1102-1112
mantissa, 1186
many-to-many relationships, 811-813
many-to-one relationships, 810-811
MAPI (Messaging API), 917, 1186
margins, reports, adjusting, 543-546
master databases, 1186
master tables, 1186
matches, records search
automatically, 186
manually, 183-185
mathematical functions, 294-295, 304-306
maximizing Access 97 document windows, 69-70
.mdb extension (Access 97), 111
.mdb files, 866
Jet 3.5, 28
.mde files, 26
creating databases, 105
MDI servers, 1186
Media Server
ActiveMovie Streaming Format (ASF), 579
features, 579
Microsoft Internet products, 579
memory
caches, 1168
shared application memory, 1198
troubleshooting, bound object controls, 693
VM (virtual memory), 1205
memos, 1187
menu macros (Access 97), 1218-1219
menus, 1187
composite menus, 1171
hierarchical menus, 1182
shortcut (Access 97), 78-79
submenus, 1200
Merchant Server
features, 578-579
Microsoft Internet products, 578-579
Secure Electronic Transactions (SET), 578-579
merge documents
country selection, 765-767
differing data source, creating, 767-770
records, building, 764-767
Merge It with MS Word command (Tools menu), 768
metafiles, 1187
methods, 1187
object behavior, accounting for, 792-793
objects, 670-671
Seek, 1198
Microsoft
Access development, 1095-1096
Access VBA development, 978-979
ASP development, 652
Internet, ActiveX technology development, 1096-1097
Internet products
dbWeb 1.1a, 579
FrontPage 97, 579-580
Index Server 1.1, 578
Internet Explorer 3.0, 576-580
Internet Information Server, 577
Media Server, 579
Merchant Server, 578-579
NetMeeting 2.0, 576-577
NetShow, 577
ProxyServer, 577-578
Internet strategy, 575-576
Internet technologies
Active Server Pages, 582
ActiveMovie, 581
ActiveX, 581
Common Object Model (COM), 580
DirectX, 582
Distributed COM, 581
ISAPI, 582
OLE DB, 582
PC97, 583
Visual Basic, 582
Microsoft Access 97 Object Library, 1026-1027
Microsoft Access command (Help menu), 408
Microsoft Access Help (Help menu), 98, 434
Microsoft DAO 3.5 Object Library, 1026-1027
Microsoft Excel 8.0 Object Library, referencing, 1106-1108
Microsoft Exchange, sending reports, 570-572
Microsoft Graph 97
charts, creating, 709-727
drawbacks
CPU speeds, 720
datasheet transfers, 720
graphs, creating, 709-727
Microsoft Office 8.0 Object Library, 1026-1027
references for CommandBar objects, 1075-1078
Microsoft on the Web command (Help menu), 91
Microsoft Outlook, sending reports, 570-572
Microsoft Query
form letters, creating, 777-778
mail merge data source, creating, 770-777
Microsoft Transaction Server (MTx), 627
MIDI (Musical Instrument Digital Interface) chunks, 1169
migrating Access 2.0 .MDB files to Access 97, 1152-1154
mini-server applications, 669
mini-servers, 1187
minimizing Access 97 document windows, 70-71
minus sign (-) operator, 290
mirroring disks, 1176
miscellaneous functions, 307-308

Oracle Ex. 1009, p. 1268
Oracle v. DeBellis
IPR2024-00221

miscellaneous operators, 292-293
MISF (Microsoft Internet Security Framework), 1187
Missing Memo File dialog box, 210
mission-critical databases, 1187
mobile computing, 955
modal dialog boxes, 1187
modeless dialog boxes, 1187
modeless windows, 1187
modeling data, 797-802
modifying
  column headers, queries, 271-273
  crosstab queries, detail level, 365-367
  graphs, design features, 715-720
  image control properties, 708-727
  numeric field data types, 157-158
  queries, records criteria, 269-270
  Query Design window, properties, 264-267
  tables
    AutoNumber field, 739-742
    field data types, 156-158
    relationships, 158-159
  text field data types, 158
  toolbar properties, 75-78
  unlinked graph properties, 714-715
  Word 97 documents with VBA code, 1118-1120
Module command (Insert menu), 1008
module languages (SQL), 837
module level scope, Access VBA variables, 990-991
Module window
  Access VBA toolbar, 1008-1010
  Auto List Members, 26
  function keys, assigning, 80-81
modules (Access VBA)
  access, 981-985
  functions, 65-66
  global modules, 1181
  reports, 981-985
  text comparisons 1020
monikers, 1187
monitors, 1188
More Controls tool (Toolbox), 446
mouse
  Access 97 shortcut menus, 78-79
  drag-and-drop, 1177
  events, 1062-1068
MoveSize method, 1072-1075
moving
  Briefcases, 967
  controls
    forms, 421-422
    reports, 526-532
  form elements, 419-426
  Recordset objects, specific records, 1037
multicolumn equi-joins
  creating, 329-332
  unique values, selecting, 329-332
multicolumn reports, 514-515
  printing as mailing labels, 547-550
multidimensional arrays, creating (Access VBA), 992-993
multifield primary keys, creating, 153
multifield sorts, records, 193-195
multiline text boxes, creating, 458-463
multimedia, 1188
multiple record values, updating, 383-387
multiple subreports, creating unbound reports, 568-569
multiple-field indexes, 154-155
multiple-tier drivers (ODBC), 918
multiplication (*) operator, 290
multiprocessing, 1188
  scalable multiprocessing, 1197
  symmetrical multiprocessing, 913
multitable queries, joining, 322-323
multitasking, 1188
  background, 1167
  foregrounds, 1180
  multithreaded preemptive, 29
multithreaded applications, 1188
multithreaded preemptive multitasking, 29
multiuser applications, 868-877
  administrating, 910-912
  databases, splitting, 873-875
  groups
    deleting, 891-892
    managing, 884-892
  opening, 877-881
  ownership, 892-895
    changing, 892-897
  preparations, 872-873
  security, 881-892
    LANs, 881
    passwords, 883-884
  users
    deleting, 891-892
    managing, 884-892
multiusers, 1188
My Briefcase icon, 969

N

Name and Organization dialog box, 32
Name command (Insert menu), 222, 1103
named arguments (Access VBA), 997
named constants, 309
named database objects (Access VBA), 993
named pipes, 1188
named range of cells, specifying (Customer.xls), 1111
names (objects)
  prefixes, 1210
  tags, 1211
naming
  Access 97 constants, 1225
  object structure, 1210-1212

Oracle Ex. 1009, p. 1269
Oracle v. DeBellis
IPR2024-00221

queries, 278-279
VBA objects, base names, 1219
naming conventions
Access VBA, 980-981
LNC, 994-995
variables, 994-995
databases, objects, 1215-1219
object collections, 1024-1025
VBA objects
tags, 1220
variables, 1227
natural joins, 1188
navigating intranets, 584-585
NBF (NetBEUI Frame), 1188
nested objects, 1188
nested procedures, 1188
nested queries, 1188
NetBEUI (NetBIOS Extended User Interface), 914
NetBIOS (Network Basic Input/Output System), 1189
NetMeeting 2.0
downloading, 576-577
features, 576-577
Microsoft Internet products, 576-577
NetShow
features, 577
information, obtaining, 577
Microsoft Internet products, 577
networking applications, 865-866
networks
Access, installing, 866-867
databases, sharing, 902-910
file sharing, 868-877
accessing, 907-910
administrating, 910-912
folders, 868
preparations, 872-873
security, 881-892
servers, 907
system folders, 868
Windows 95, 902-904
groups
deleting, 891-892
managing, 884-892
intranets, 1184
ISDN (Integrated Services Digital Network), 1184
LANs (local area networks), 1185
security, 881-882
users
deleting, 891-892
managing, 884-892
VPN (Virtual Private Network), 1205
WANs (wide area networks), 1205
New command (File menu), 113, 160
New Database button, 207
New Database command (File menu), 207, 732
New Database dialog box, 732
New dialog box, 113, 959
New Form dialog box, 399
New Query dialog box, 312, 747
New Record button, 169
New Report dialog box, 516, 568
New Style Name dialog box, 413
New Table dialog box, 130
newline pairs, 1189
newspaper columns, mailing labels, 550
Next Character keyboard option, 168
Next Field keyboard option, 168
Next Record keyboard option, 168
NFS (Network File Server), 1189
nilTestQuery function, 1041-1044
creating, 1042-1044
nonclustered indexes, 827-828, 1189
normal forms, 1189
normalization, 1189
fifth normal form rule, 807-808
first normal form rule, 803
fourth normal form rule, 806-807
relational databases, 802-813
second normal form rule, 805
third normal form rule, 805-806
Northwind.mdb database, copying, 162-163
not equal theta joins, creating, 344-346
Not operator, 171
not-equal joins, 1189
notebook computers, 955
NT (New Technology), 1189
NTFS (New Technology File System), 1189
null values, display format, 124-125
null variables, 1189
Number of Update Retries option, multiuser applications, 873
numeric data
field size property, selecting, 120-123
modifying, 157-158
size property listing, 121-122
numeric fields (Excel 97)
double precision, 734-735
single precision, 734-735
numeric literals, 293
numeric values, extracting from OLE objects, 759-760

O

Object Browser, 26
Excel programmable objects, referencing, 1106-1108
implementing, 1027-1028
launching, 1027-1028, 1054-1056
viewing class modules, 1054-1056
Object Browser command (Tools menu), 1009
Object Browser command (View menu), 1132
Object Browser dialog box, 81

Oracle Ex. 1009, p. 1270
Oracle v. DeBellis
IPR2024-00221

object collections
argument method, 1030
connections, 1024-1025
creating references, 1030
defined, 1023-1024
documents, 1024-1025
forms/reports, 1024-1025
index method, 1030
literal method, 1030
naming conventions, 1024-1025
Object command (Insert menu), 703
object frames, programming (Word 97), 1120-1122
object libraries
Microsoft Access 97 Object Library, 1026-1027
Microsoft DAO 3.5 Object Library, 1026-1027
Microsoft Office 8.0 Object Library, 1026-1027
programmatic references, 26
Visual Basic for Applications 5.0, 1025-1027
object models, displaying (Word 97), 1116-1118
object variables
creating, 1032-1033
Debug window, implementing, 1033-1034
object-enabled applications, 671
object-oriented design databases, 790-793
objects, 670-673, 1189
advantages, 671
aggregate objects, 1164
base names, 1211-1212
compound, 1214
creating, 1212-1215
field, 1214
length limits, 1213
naming rules, 1212-1215
ordering, 1214-1215
qualifiers, 1211-1212
suffixes, 1212
behavior, accounting for, 792-793
classes, 672-673
code, 1190
collections, 1170
colors, changing, 413-421
COM (Component Object Model), 1171
containers, 673-681
Control objects, 671
DAOs (data access objects), 681, 1173
data entities, 791-792
database objects, 671, 1174
DBEngine, 681
drag-and-drop, 1177
embedded objects, 1178
enabled objects, 1178
Form objects, 671
frames, 1190
grouping (LNC), 1209-1210
groups, 1181
hierarchies, 672-673
in-place activation, 680-681
inheritance, 672-673
instances, 1183
integrating relational databases, 787-788
Leszynski Naming Conventions, 1208-1209
libraries, 1190
linked objects, 1185
menu macros, 1218-1219
methods, 670-671
monikers, 1187
names
prefix, 1210
structuring, 1210-1212
tags, 1211
naming conventions for databases, 1215-1219
nested objects, 1188
object-enabled applications, 671
OLE Automation, 681-682
OOP (object-oriented programming), 671
opening hyperlinks, 592-594
ownership, 892-895
changing, 892-897
permissions, 892, 1190
granting, 895-902
revoking, 895-902
persistent objects, 1192
printer objects, 1193
programmable objects, 1193
properties, 670-671
dynamic, 791
reading, 1031-1032
setting, 1030-1032
static, 791
pseudo-objects, 1194
QueryDef objects, 671
RAOs (Remote Automation Objects), 1196, creating, 670
referencing, 985-986
Remote Data Object, 1196
Report objects, 671
screen objects, 1197
selecting, 1198
system objects, 1201
tags, character length, 1211
toggles, 1202
types, 672-673
obtaining
Common Object Model (COM) specifications, 580, 1096-1097
context-sensitive help, 90-91
Distributed COM working specifications, 581
Excel 97 example worksheet, 731-732
LNC complete text, 1229
NetShow information, 577
QBE summary information, 351-358
ODBC (Open Database Connectivity), 914-918, 1190-1191
Administrator tool, creating, 924-929
advanced database options, 86-88
APIs, 917
CLI (Call Level Interface), 919
databases, 935-940
data source
Access 97, creating, 637-642
default, specifying, 642-643
drivers, 918-920
installing, 920-923

Oracle Ex. 1009, p. 1271
Oracle v. DeBellis
IPR2024-00221

DTCs (Distributed Transaction Controllers), 917
MAPI (Messaging API), 917
queries, logging, 944-945
TAPI (Telephony API), 917
worksheets, linking, 228-233
WOSA (Windows Open Services Architecture), 917
ODBC Create Data Source Administrator dialog box, 928
ODBC Databases( ) function, 945
ODBC Microsoft Access 97 Setup dialog box, 640, 773
ODBC SQL Server Setup dialog box, 925
ODBCDirect, 24, 1023, 1190
overview, 1044-1047
ODE (Office 97 Developer Edition),
components, installing, 44-45, 1135-1136
run-time versus retail access, 1136-1137
Office Assistant, 97-102
activating, 98-102
appearance, 97-98
characters, selecting, 100
help topics, searching, 98-102
options, setting, 100-102
text balloons, 97-98
tips, 98
Office Links command (Tools menu), 201, 525, 768
offsets (reference points), 1190
OLE (Object Linking and Embedding), 665-668
32-bit OLE, 665
32-bit OLE Controls (.ocxs), 31
activations, 1163
ActiveX, 665
ActiveX Controls, 665, 669, 682
automation, 1166
automation components, 1097-1099
Clipboard, 666
COM (Component Object Model), 665
compliant applications, 668-670
compound document, 1172
containers, 673-681
DCOM terminology, 1097
DDE (dynamic data exchange), 667
compared, 675-676
destination documents, 1175
developmental history, 1095-1096
functions, 956-957
in-process, 1183
IPC (interprocess communication), 666
links, creating, 760-762
MDI servers, 1186
Microsoft Internet technologies, 582
objects
data type fields, 122-123
in-place activation, 680-681
Personnel Actions, displaying, 693-696
values, extracting, 759-760
out-of-process, 1191
Registry, 682-689, 686-687
running states, 1197
SDI servers, 1197
source documents
embedding, 676-679
linking, 676-679
support, 31
OLE 2.0, naming standards for objects, 1024-1025
OLE Automation, 1190
OLE controls, 1190
Calendar Control, 1124-1131
registering, 1123-1124
OLE DB, 1191
OLE DLL, 1191
OLE servers, Excel 97, implementing, 754-762
one-to-many relationships, 810
one-to-one relationships, 809-810
Online VBA Help File (Word 97), 1116-1118
OOP (object-oriented programming), 671
Open command (File menu), 163
Open Database command (File menu), 50, 206
Open dialog box, 49-50, 169
open forms, referring (Access VBA), 1087-1088
open queries, Access 97 functions, 66
OpenDoc, 1191
OpenForm method, 1072-1075
opening
Access 97 databases, 50-51
Customer.xls, worksheet objects, 1109-1111
databases, 48-50
hyperlinks to Access objects, 592-594
Recordset objects, 1034-1040
shared databases, 877-881
OpenModule method, 1072-1075
OpenQuery method, 1072-1075
OpenReport method, 1072-1075
operands, 171, 1191
operating modes
Design, 66-67
Run, 66-67
Startup, 66-67
operating systems, 1191
event-driven, 1179
UNIX, 1204
operators, 1191
arithmetic, 289-290
arithmetic operators, 171
assignment, 290-291
categories, 289
comparison operators, 171
concatenation, 291
consignment, 290-291
division (/), 290
expression element, 288-289
expressions, 171

Oracle Ex. 1009, p. 1272
Oracle v. DeBellis
IPR2024-00221

greater than (>), 291
greater than or equal to (>=), 291
identifier, 292
Jet SQL, 840-842
less than (<), 291
less than or equal to (<=), 291
logical, 291
logical operators, 171
minus sign (-), 290
miscellaneous, 292-293
multiplication (*), 290
operands, 171
plus sign (+), 290
relational operators, 1195
optimistic locking, 1191
Option Button tool (Toolbox), 445
option buttons, 1191
forms, adding, 464-472
Option Group tool (Toolbox), 445
Option Group Wizard, 465-469
optional arguments (Access VBA), 997
options
Help windows, 97
setting Office Assistant, 100-102
Options command (Tools menu), 81, 112, 167, 410, 872
Options dialog box, 82
Or operator, 171
order by, table properties, 115
ORDER BY clause (SQL), 839
ordering objects by base names, 1214-1215
ORK (Office 97 Resource Kit), installing, 44
ORM (Object Role Modeling), 801
out-of-process (OLE), 1191
outer joins, 323, 1191
tables, creating, 340-342
outer queries, 1191
Outlook, 25
output all fields property
queries, formatting, 347-349
selecting, 350-351
OutputTo method, 1072-1075
overriding properties in fields, 509-510
OWNERACCESS reserved word (Jet SQL), 841
ownership
objects, changing, 892-897
permissions, 892
granting, 895-902
revoking, 895-902

P

Page Break tool (Toolbox), 446
page breaks, controlling in reports, 559
Page Header/Footer command (View menu), 557
Page Setup command (File menu), 60, 275, 544, 783
Page Setup dialog box, 60, 275, 544
Pages command (View menu), 525
Paint 95, graphics, displaying, 56-57
PAL, comparisons with Access VBA, 979-980
palettes, color, 1170
Paradox
files, identifying, 206
key fields, establishing, 213
tables, modifying, 863
parameters, 1192
adding
make-table queries, 279-284
queries, 359-360
data type queries, 360
designing queries, 358-361
selecting with Query Wizard, 773-777
specifying data types with queries, 360-361
Parameters command (Query menu), 360
parsing, 1192
Partial ReplicationWizard, 31, 971
installing, 972
obtaining, 972
replicating, 971-973
pass-through queries (SQL), 940-945
connect string property, setting, 942-944
executing, 940-942
Password Required dialog box, 208, 883
passwords
authentication, 1166
databases, 883-884
establishing, 884
Paste Append command (Edit menu), 170
Paste As dialog box, 702
Paste button, 170
Paste command (Edit menu), 73, 160, 165, 170, 407, 433, 524
Paste Errors table, 249
Paste Special command (Edit menu), 678
Paste Table As dialog box, 160, 333, 746
pasting tables (Clipboard), 159-160
PC databases, file formats, identifying, 205
PC97, Microsoft Internet technologies, 583
.pcd files, 700
.pcx files, 699
PDCs (Primary Domain Controllers), 1192
peer-to-peer networks, sharing (Access 97), 867
Performance Analyzer Wizard, 30
permissions, 1192
database workgroups, 902-910
explicit permissions, 896
full permissions, 896
groups, changing, 897-899
objects, 892, 1190
granting, 895-902
revoking, 895-902
users, specific, 900-901
Permissions dialog box, 902

Oracle Ex. 1009, p. 1273
Oracle v. DeBellis
IPR2024-00221

persistent graphics, 1192
persistent objects, 1192
Personnel Actions
candidate fields, 140-151
creating logos, 702-705
creating tables, 141-144
default field values table, 1232
design elements, 138-139
designing, 138-141
employee pictures, displaying, 693-696
entries table, 1233
field information, assigning, 140-151
field properties table, 1232
graphic images, including, 692-693
key fields table, 1233
records, appending, 433-436
tables, information criteria, 139-140
validation criteria table, 1232-1233
pessimistic locking, 1192
PINs (Personal ID Numbers), 884
PIVOT reserved word (Jet SQL), 841
placeholders
custom format, 125-127
input masks, 128-129
plus sign (+) operator, 290
.png files, 700
pointers, 1192
points, 1192
pokes, 1193
populating combo boxes (Query Builder), 484-490
PPP (Point-to-Point Protocol), 1193
PPTP (Point-to-Point Tunneling Protocol), 1193
precedence, 1193
prefixes
combinations, VBA objects, 1225
LNC, procedure names, 1226
object names, 1210
VBA objects, variables, 1224-1225
preserving variable duration, 990-991
previewing form letters (Mail Merge Wizard), 764-767
primary keys, 1193
relational databases, 796
selecting tables, 152-153
setting, 213-217
tables
creating, 152-153
design grids, 116-117
primary verbs, 1193
Print command (File menu), 60, 72, 433, 544
print events, 1062-1068
print formats (Access 97), 61-62
Print Preview command (File menu), 59, 72, 407, 433, 523
Print Topic command (Help menu), 97
print zone, 1193
printer objects, 1193
Printer Setup dialog box, 514
printing
contents of tables, 59-62
multicolumn reports as mailing labels, 547-550
query reports, 273-276
reports, 543-546
footers, 559
headers, 559
to Debug window, 1018-1019
PRIVILEGES reserved word (ANSI SQL), 847
Procedure command (Insert menu), 1021
Procedure Definition command (View menu), 1011
procedure names (LNC)
prefixes, 1226
tags, 1226
procedures, 1193
Access VBA, 983-985
invoking Access VBA, 983
nested procedures, 1188
stored procedures, 1199
subprocedures, 1200
process servers, 1193
processing SQL statements, 284-286
program flow
branching, 998
conditional execution (Access VBA), 998
Go To statements (BASIC), 998
repetition (Access VBA), 998
termination statements (Access VBA), 998
programmable objects, 1193
manipulating, 1102-1112
referencing, 1106-1108
VBA hierarchy, 1103-1104
programmatic references to object libraries, 26
programming
Calendar Control, 1127-1131
initializing, 1183
objects, 670-673
classes, 672-673
hierarchies, 672-673
inheritance, 672-673
OOP (object-oriented programming), 671
types, 672-673
Word 97 documents in object frames, 1120-1122
Programs menu commands, Start, 48
Progress Monitor, 40
projection, 1194
properties, 1194
Arrow Key Behavior, 163
converting line graphs, 717-720
description, tables, 115
fields, overriding, 509-510
filters in tables, 115
modifying
image controls, 708-727
Query Design window, 264-267
toolbars, 75-78
unlinked graphs, 714-715
objects, 670-671

Oracle Ex. 1009, p. 1274
Oracle v. DeBellis
IPR2024-00221

order by tables, 115
setting
objects, 1030-1032
unbound object frames, 705-707
sizing unbound object frames, 705-707
vaildation rule in tables, 115
validation text in tables, 115
Properties button, 173
Properties command (File menu), 668
Properties command (View menu), 57, 137, 408, 524
Properties window, 1194
property values, reading, objects, 1031-1032
protocols, 1194
DHCP (Dynamic Host Configuration Protocol), 1176
ECP (Extended Communication Protocol), 957
IP (Internet Protocol), 1184
hosts, 1182
IPX/SPX (Internetwork Packet Exchange/ Sequenced Packet Exchange), 914, 1184
NetBEUI (NetBIOS Extended User Interface), 914
NetBIOS (Network Basic Input/Output System), 1189
PPP (Point-to-Point Protocol), 1193
PPTP (Point-to-Point Tunneling Protocol), 1193
stacks, 1194
TCP/IP (Transport Control Protocol/Internet Protocol), 914, 1201
prototype declarations, Win32 API Viewer, 1147-1150
ProxyServer
downloading, 577-578
features, 577-578
Microsoft Internet products, 577-578
pseudo-objects, 1194
public level scope, Access VBA variables, 990-991
Publish to the World Wide Web Wizard, launching, 643
punctuation (SQL), 849-850

Q

QBE (Query By Example), 835-837
SQL, 837
summary information, obtaining, 351-358
qualifications, 1194
qualifiers, base names, 1211-1212
queries, 1194
action, 279-280, 371
action queries, syntax, 860-861
ASP, exporting, 655-660
calculating field values for expressions, 316-319
charts, creating, 709-711
column headers, renaming, 271-273
converting, 280-282
creating, 262-279
form letters, 777-778
forms, 452-454
Web pages, 603-606
creation criteria for expressions, 312-314
crosstab, 279-280
creating, 361-367
data entry validation expressions, 310
display, sorting, 267-269
dynamic Web pages, creating, 643-651
exporting templates, 608-610
Expression Builder, addition criteria, 314-316
expressions, defined, 288-289
filters, saving as, 197-202
formatting, output field properties window, 347-349
HTML, exporting, 597-600
indirect table relationships, creating, 326-329
linking graphs, 723-727
modifying column headers, 271-273
naming, 278-279
nested queries, 1188
ODBC, logging, 944-945
outer queries, 1191
parameters
adding, 359-360
data types, 360
designing, 358-361
troubleshooting, 332
print file formats, 273-276
properties
appending criteria, 346-347
updating criteria, 346-347
records criteria
modifying, 269-270
selecting, 267-269
reports, printing, 273-276
reserved words (SQL), 285-286
saving, 278-279
select, 279-280
selecting fields, 264-267
setting criteria expressions, 310-316
specifying sort order, 326
SQL
pass-through queries, 940-945
SELECT queries, 847-861
subqueries, 858-859, 1200
tables, editing, 271
transferring data, 276-278
troubleshooting, 319
updatable, troubleshooting, 349
Query Builder, 447
combo boxes, populating, 481
Query by Example, *see* QBE
Query Design window properties, modifying, 264-267

Oracle Ex. 1009, p. 1275
Oracle v. DeBellis
IPR2024-00221

Query menu commands
Append Query, 379
Crosstab Query, 362
Delete Query, 382
Make Table Query, 374, 737, 743
Parameters, 360
Show Table, 744
SQL-Specific, 941
Update Query, 386, 742
Query Parameters dialog box, 361
Query Properties sheet (SQL), 942
Query Properties window, external databases, querying, 367-369
Query Wizard, database parameters, selecting, 773-777
QueryDef objects, 671, 1029, 1194
creating, 1040-1041
Recordset objects, emulating, 1039-1040
querying
external databases, 367-369
indexes, 828-830

R

RAID (redundant array of inexpensive disks), 1194
ranges, database groupings, 554-572
RAOs (Remote Automation Objects), creating, 670
RAS (Remote Access Services), 955
RDBMS (Relational Database Management System), 203, 1194
advantages over spreadsheet functions, 110-111
elements, 110-111
Excel 97 data, processing, 730-754
relations, 1195
RDO (Remote Data Object), 24
reading object property values, 1031-1032
REAL reserved word (Jet SQL), 841
rearranging
forms, 426-431
tables in field sequences, 155-156
reconciliation, replica sets, emulating, 967-971
reconciliation handlers, binding, 956
records, 1194
adding tables, 168-170, 175-177
appending tables, 170, 433-436
building merge documents, 764-767
combo boxes, searches, 486-489
current records, 1173
deleting
Form view, 437
referential integrity, 383
tables, 170
editing, 436
exporting
Clipboard, 253-257
Excel 97, 754
tables, 253-257
as text files, 255-257
expressions, sorting, 555
filtering
advanced filters, 193
composite filters, 195-197
compound filters, 193-195
copying, 201-202
exporting, 201-202
by form, 189-192
loading, 197-202
mailing, 201-202
saving as queries, 197-202
by selection, 187-189
tables, 186-199
grouping
by category, 552-553
databases, 551-555
by range, 554-572
HTML tables, importing, 618-623
importing Clipboard, 247-250
matches
finding, 183-185
finding automatically, 186
modifying criteria for queries, 269-270
refreshing, 442, 1195
replacing
Clipboard, 250-251
tables, 170
selecting tables, 170
selecting criteria queries, 267-269
sorting
advanced sorts, 193
databases, 551-555
exporting, 201-202
mailing, 201-202
multifield sorts, 193-195
multiple fields, 182
single field, 181
subforms, 439-442
tables, 179-183
tables
appending, 378-380
deleting, 381-383
Records menu commands
Filter, 73, 187-188, 433
Go To New, 74
Refresh, 442
Remove Filter/Sort, 197, 248
Save Record, 169, 435
Sort, 73, 181, 433
Recordset objects, 1030
emulating QueryDef object, 1039-1040
intrinsic constants, 1035
opening, 1034-1040
specific records, moving, 1037-1038
re-creating example worksheet, 746-750
Rectangle tool (Toolbox), 446
recursions, 1195
redirectors, 1195
Redo command (Edit menu), 1009
redundant fields, deleting, 253

Oracle Ex. 1009, p. 1276
Oracle v. DeBellis
IPR2024-00221

reference points, offsets, 1190
references, 1195
data access objects, creating, 1025-1027
object collections, creating, 1030
References command (Tools menu), 112, 985, 1011, 1131
referential integrity, 1195
enforcing, 834
tables, 151-152
records deletion, 383
troubleshooting, 389
referring (Access VBA)
open forms, 1087-1088
report forms, 1087-1088
Refresh command (Records menu), 442
Refresh Interval option, multiuser applications, 873
refreshing records, 442, 1195
registering OLE controls, 1123-1124
Registry, 682-689, 1195
entries, 928
files, replacing with backups, 687-689
manipulating automation servers, 1108-1109
OLE, 686-687
Registry Editor, 683-686
subtrees, 684
relation objects, 1030
relation tables (relational databases), 796
Relational Database Management Systems, *see* **RDBMS**
relational databases, 787
designing, 790
keys, 796-797
normalization, 802-813
objects, integrating, 787-788
tables, 796-797
relational operators, 1195
relationships
entities, 809-818
establishing
cascading deletions, 388-389
cascading updates, 388-389
tables, 376-377, 750-753
many-to-many relationships, 811-813
many-to-one relationships, 810-811
modifying tables, 158-159
one-to-many relationships, 810
one-to-one relationships, 809-810
Relationships command (Tools menu), 148, 159, 376
Relationships dialog box, 157, 376
tables, new versus existing, 148-152
relative file paths, hyperlinks, 584
Remote Automation Object, 1196
Remote Data Object, 652, 1196
overview, 1044-1047
Remove Filter/Sort command (Records menu), 197, 248
Rename command (Edit menu), 279, 940
Rename method, 1073-1075
renaming column headers, 272-273
RepaintObject method, 1073-1075
repairing
database files, 911
databases, 104-105
Replace command (Edit menu), 84, 186
Replace dialog box, 80-81
Replace in Field dialog box, 186
replacing
AddMenu macro actions (CommandBar objects), 1075-1078
AutoExec macros (custom CommandBar), 1079
records
Clipboard, 250-251
tables, 170
replica sets
creating ContactData.mdb, 962-965
reconciliation, emulating, 967-971
replicating
Briefcases, 956-957
contact management applications, 958-962
replica sets, 962-965
synchronizing, 965-966
updating, 957-966
databases, 971-973
replication, 1196
briefcases, 31, 1168
Replication command (Tools menu), 957
report forms, referring (Access VBA), 1087-1088
Report Header/Footer command (View menu), 557
report level scope, Access VBA variables, 990-991
report modules
Access VBA, 981-985
implementing Error event, 1007
Report objects, 671
Report Print Preview mode, toolbar buttons, 525
Report Wizard, 447, 514
blank reports, creating, 555-564
hint screens, 518
overview, 551
reports
creating, 516-523
modifying, 526-543
subreports, 555-557
reports
adding
bound object controls, 692-701
linked subreports, 564-568
toolbox controls, 569-570
unbound object frames, 701-707
by category, creating, 559-564
calculated controls, adding, 534-539

Oracle Ex. 1009, p. 1277
Oracle v. DeBellis
IPR2024-00221

controls
calculated controls, 534-539
deleting, 526-532
editing, 526-532
formatting, 540
relocating, 526-532
creating, 516-523
Report Wizard, 555-564
customizing
AutoFormats, 526
fields, selecting, 517
footers
adding, 557-559
deleting, 557-559
printing, 559
forms, compared, 513-514
grouping, 516-523
groups/totals reports, 515
headers
adding, 557-559
deleting, 557-559
printing, 559
HTML, exporting, 610-613
inventory reports, creating, 516
Keep Together property, 546-547
line spacing, adjusting, 541-543
mailing labels, 515
newspaper columns, 550
margins, adjusting, 543-546
Microsoft Exchange, sending, 570-572
Microsoft Outlook, sending, 570-572
modifying in Report Wizard, 526-543
multicolumn reports, 514, 515
page breaks, controlling, 559
printing, 543-546
queries, 273-276
record source, changing, 534
single-column reports, 514
SQL statements, 863-864
subreports, 1200
tabular reports, 515
types, 514-515
unbound reports, 515
windows, 523-526
repositioning Help windows, 92-93
Requery method, 1073-1075
reserved words, 1196
ANSI SQL
AUTHORIZATION, 847
BEGIN r, 847
CHECK r, 847
CLOSE r, 847
COMMIT r, 847
CREATE VIEW r, 847
CURRENT r, 847
CURSOR r, 847
DECLARE r, 847
DROP VIEW r, 847
FETCH r, 847
GRANT r, 847
PRIVILEGES r, 847
REVOKE r, 847
ROLLBACK r, 847
TRANSACTION r, 847
VALUES r, 847
WORK r, 847
Jet SQL, 840-842
(date specifier), 842
<> (not equal), 842
? (LIKE wild card), 842
BINARY, 841
BOOLEAN, 841
BYTE, 841
CURRENCY, 841
DATETIME, 841
DISTINCT, 845
DISTINCTROW, 841, 845
DISTINCTROW SQL, 842
DOUBLE, 841
INT[EGER], 841
LONG, 841
LONGBINARY, 841
LONGTEXT, 841
OWNERACCESS, 841
PIVOT, 841
REAL, 841
SHORT, 841
TRANSFORM, 842
VARCHAR, 841
SQL, 839
queries, 285-286
Reset command (Run menu), 1009
resetting (default) toolbars, 449
responding
Access VBA data events, 1090-1094
event functions, 1060-1061
restoring databases, 910-912
restrictions, 1196
retail access versus run-time access (Office 97 Developer Edition), 1136-1137
Return Data to Microsoft Word command (File menu), 776
REVOKE reserved word (ANSI SQL), 847
revoking permissions, 895-902
RGB (red, green, blue) color scheme, 1196
RIFF (Resource Interchange File Format) chunks, 1169, 1196
RISC (reduced instruction set computing), 913
ROLLBACK reserved word (ANSI SQL), 847
rollbacks, 1196
routines, 1196
Row Height command (Format menu), 200
rows, 1197
dynasets, 1178
RPCs (remote procedure calls), 1196
RSIs (repetitive stress injuries), 162
rulers (Form Design window), 405
rules, 1197
normalization, 802-808
fifth normal form rule, 807-808
first normal form rule, 803
fourth normal form rule, 806-807
second normal form rule, 805
third normal form rule, 805-806
Run command (Start menu), 32
Run dialog box, 32

Oracle Ex. 1009, p. 1278
Oracle v. DeBellis
IPR2024-00221

Run menu commands
Clear All Breakpoints, 1017
End, 1009
Reset, 1009
Run mode, 66-67, 1197
controls, appearance, 450-451
Run to Cursor command (Debug menu), 1011
run-time versus retail access (Office 97 Developer Edition), 1136-1137
run-time Access, distributing, 867
run-time errors
Err function, detecting, 1006-1007
handling, 1005-1007
RunCode method, 1073-1075
RunCommand method, 1073-1075
RunMacro method, 1073-1075
running states (OLE), 1197
RunSQL method, 1074-1075

S

SAM (Security Accounts Manager), 1197
sample .htx file, 634-662
sample .idc files, 631-634
sample.htx file source code, 634-635
sample2 .idc files, 631-634
Save As command (File menu), 279
Save As dialog box, 135, 144, 255
Save As HTML command (File menu), 614, 643
Save As Query command (File menu), 197
Save As/Export command (File menu), 597, 655
Save command (File menu), 72, 79, 407, 433, 523
Save Copy As command (File menu), 705
Save Record command (Records menu), 169, 435
saving
filters as queries, 197-202
queries, 278-279
scalable applications, creating, 913-914
scalable multiprocessing, 1197
scaling graphic images (Access 97), 696-699
scope levels, 1197
Access VBA variables, 990-991
form, 990-991
global, 990-991
local, 990-991
module, 990-991
public, 990-991
report, 990-991
arrays, 993
scope prefixes, VBA objects, 1224-1225
screen objects, 1197
scripting ActiveX, 1164
scroll bars, 1197
Form Design window, 406
SCSI-2 drives for data marts, 626-627
SCSI-3 drives for data marts, 626-627
SDI servers, 1197
searches, records in combo boxes, 486-489
searching help topics (Office Assistant), 98-102
second normal form rule, 1197
normalization, 805
Secure Electronic Transactions (SET), Merchant Server, 578-579
security
Access 97 functions, 66
encryption, 911-912
file sharing, 881-892
ownership, 892-895
user-level security, 904-906
groups, managing, 884-892
networks, LANs, 881-882
passwords, 883-884
permissions in objects, 892
SAM (Security Accounts Manager), 1197
SID (security ID), 1199
users, managing, 884-892
Security command (Tools menu), 66, 883
Security Wizard, ownership, changing, 894-895
Seek method, 1198
SELECT * FROM table name (Access SQL), 350
Select a Jet Database dialog box, 1148
Select All command (Edit menu), 540
Select All Records command (Edit menu), 276, 379
Select Case...End Select constructs, 1000-1002
SELECT command (SQL), 839
Select Data Source dialog box, 638, 943
Select Entire Field keyboard option, 168
Select Form command (Edit menu), 420, 423, 428
select lists, 1198
Select Microsoft Word Document dialog box, 768
Select Objects tool (Toolbox), 445
SELECT queries, 847-861
select queries, 279-280
building, 372-373
make-table queries, converting, 374-375
Select Report command (Edit menu), 536, 559
Select Workgroup Information File dialog box, 876
Selected Axis command (Format menu), 717
Selected Chart Title command (Format menu), 716
selected table records, executing calculations, 356-359

Oracle Ex. 1009, p. 1279
Oracle v. DeBellis
IPR2024-00221

selecting
controls in forms, 421-422
data in tables, 53-56
display format for data types, 123-128
field data types, tables, 119-120
fields
output all fields property, 350-351
queries, 264-267
filtering records, 187-189
form elements, 419-426
forms, 398
multicolumn equi-joins, unique values, 329-332
numeric data, field size property, 120-123
objects, 1198
Office Assistant characters, 100
queries, records criteria, 267-269
Query Wizard database parameters, 773-777
records for tables, 170
size of tables, 119-120
tables, primary keys, 152-153
text
fields, 164-165
grids, 164-165
text boxes, 164-165
text data, field size property, 120-123
self-joins, 323, 1198
tables, creating, 342-344
Send command (File menu), 570
Send dialog box, 570
sending
data via Word 97 automation server, 1116-1123
reports
Microsoft Exchange, 570-572
Microsoft Outlook, 570-572
SendKeys method, 1074-1075
SendObject method, 1074-1075
separators, 1198
sequential access files, 1198
Series in Rows command (Data menu), 723
server framework for ActiveX, 1164
server-only applications, 668
server-side scripting, 652
servers, 1198
Access, installing on, 866
automation servers, 1166
clustering, 1170
Dial-Up Networking Server, 958
file sharing, 907
full servers, 1181
MDI servers, 1186
mini-servers, 1187
NFS (Network File Server), 1189
process servers, 1193
SDI servers, 1197
sessions, 1198
Set Database Password dialog box, 883
SetMenuItem method, 1074-1075
setting
Access 97
Datasheet View options, 88-89
default options, 81-89
system default options, 82-88
expressions, query criteria, 310-316
fields, default values, 145-146
fixed-width text fields for tables, 122
object properties, 1030-1032
Office Assistant options, 100-102
Personnel Actions Table, field properties, 141-144
primary keys, 213-217
system default options
advanced options, 86-88
edit/find options, 84-85
general option, 83-84
keyboard options, 85-86
view option, 83
tables
design grids, 116-117
field properties, 117-118
properties, 115
unbound object frame properties, 705-707
Setup dialog box, 33
Setup Wizard
creating distribution disks, 1138-1147
distribution disk images, 1138-1150
SetValue method, 1074-1075
SetWarnings method, 1074-1075
SGML (Standardized General Markup Language), 1182
shared application memory, 1198
shared locks, 1198
sharing
data, 1173
files on networks, 868-877
Shift+Delete key combination, 166
Shift+Enter key combination, 167
Shift+F8 key combination, 165
Shift+Insert key combination, 166
SHORT reserved word (Jet SQL), 841
shortcut keys, 1199
Access VBA modules, 1010
see key combinations
shortcut menus (Access 97), 78-79
shortcuts, icons, creating, 880
Show Next Statement command (Debug menu), 1011
Show Table command (Query menu), 744
Show Table dialog box, 156, 193, 263, 440, 744, 842
ShowAllRecords method, 1074-1075
SID (security ID), 1199
single column equi-joins, creating, 323-326

Oracle Ex. 1009, p. 1280
Oracle v. DeBellis
IPR2024-00221

single precision numeric fields, 734-735
single primary key values, table lookup fields, 332-333
single-column reports, 514
single-stepping, 1199
single-tier drivers (ODBC), 918
sites
ActiveX, 1096
JobForum, 635-637
Macmillan Publishing USA Information SuperLibrary, 731
World Wide Web Consortium (W3C), 600
size (tables)
field properties, 117-118
selecting, 119-120
Size command (Format menu), 422
sizing
controls, forms, 421-422
footers in forms, 421
headers in forms, 421
Help windows, 92-93
unbound object frame properties, 705-707
sizing handles, 1199
SMBs (Server Message Blocks), 1199
SMP (symmetrical multiprocessing), 913
Sort command (Records menu), 73, 181, 433
sort order, query result sets, 326
Sort Order dialog box, 774
sorting
data expressions, 551-555
query display, 267-269
records
advanced sorts, 193
copying, 201-202
exporting, 201-202
mailing, 201-202
multifield sorts, 193-195
multiple fields, 182
single field, 181
subforms, 439-442
tables, 179-183
Sorting and Grouping command (View menu), 524
Sorting and Grouping dialog box, 546, 551
source codes
creating Excel workbook objects (Access VBA), 1113-1114
data event handlers (VBA), 1092-1093
dbsamp2.htm (IDC), 632-633
event-handling code for macro conversion process (VBA), 1083-1086
event-handling code for main switchboard (VBA), 1053-1054
HTML .asp file (ASP), 658-660
sample.htx file (IDC), 634-635
Suppliers (unformatted) file (HTML), 602-603
VBA validation subprocedure, 1091
Source command (View menu), 600
source documents, 1199
embedding, 676-679
linking, 676-679
"spaghetti code," Go To statements, 999
specific records, moving (Recordset objects), 1037
specifying
custom CommandBar startup properties, 1079
Customer.xls, named range of cells, 1111
.idc files, default ODBC data source, 642-643
intranets, hyperlinks, 590-592
queries, parameter data types, 360-361
sort order for query result sets, 326
URLs, hyperlinks, 594-595
Spelling command (Tools menu), 72, 407, 433, 524
split-database method, 1154-1155
splitting databases, file sharing, 873-875
spreadsheets
importing, 220-233
linking, 220-233
table conversion strategy, 735-737
SQL (Structured Query Language), 835-837, 1199
aggregate functions, 308-309, 351-352, 852-855
ANSI, 836
keywords, 840
CCL (Cursor Control Language), 839
clauses, 1169
DCL (Data Control Language), 839
DDL (Data Definition Language), 801, 838
direct invocation, 837
DML (Data Manipulation Language), 838, 860-861
DQL (Data Query Language), 838
embedded SQL, 837
history, 837-838
Jet SQL
functions, 840-842
operators, 840-842
reserved words, 840-842
joins, creating, 855-857
keywords, 839
module languages, 837
pass-through queries, 940-945
connect string property, 942-944
executing, 940-942
punctuation, 849-850
QBE (query-by-example), 835-837
queries, reserved words, 285-286
Query Properties sheet, 942
reserved words, 839
SELECT queries, 847-861
statements
forms, 863-864
macros, 863-864
queries, 850-852
reports, 863-864
subqueries, 858-859
symbols, 849-850

Oracle Ex. 1009, p. 1281
Oracle v. DeBellis
IPR2024-00221

table records, calculations, 352-356
TPL (Transaction Processing Language), 838
TRANSACT-SQL, 1203
translating graphical QBE expressions, 284-286
UNION queries, 857-858
SQL command (View menu), 198, 357, 753
SQL DDL (dynamic data library)
statements
CONSTRAINT, 846
CREATE TABLE, 846
SQL Enterprise Manager, 946
SQL Server
client/server databases, 915
data sources
adding, 924-935
creating, 924-931
removing, 924-935
Distributed Transaction Controller (DTC), 627
installing, 920-923
SQL Enterprise Manager, 946
SQL Server Browser utility, 951-953
SQL Server Browser, 947, 951-953
SQL Server Login dialog box, 934, 943
SQL View command (View menu), 843
SQL-Specific command (Query menu), 941
SQLExecDirect( ) function, 944
square brackets (SQL), 849
Start command (Programs menu), 48
Start menu commands
Documents, 668
Find, 940
Run, 32
Startup dialog box, 31
startup mode, 66-67
startup properties, specifying (custom CommandBar), 1079
statements, 1199
current statements, 1173
expressions, 171
SQL
forms, 863-864
macros, 863-864
queries, 850-852
reports, 863-864
SQL DDL
CONSTRAINT, 846
CREATE TABLE, 846
static properties for databases, 791
static value lists, creating, 483-486
static variables, 1199
static Web sites, creating (Web Publishing Wizard), 614-618
StdDev( ) SQL function, 842
StdDevP( ) SQL function, 842
Step Into command (Debug menu), 1009
Step Over command (Debug menu), 1009
StopAllMacros method, 1074-1075
stored procedures, 1199
storing Access 97 library references, 1158-1159
stretch option (Bound Object Frame Properties window), 696-699
string identifier characters, 303-304
strings, 1200
binary, 1167
striping disks, 1176
Structured Query Language, *see* **SQL**
structures, 1200
structuring object names, 1210-1212
stubs, 1200
styles, 1200
subform controls (Access VBA), 1089-1094
Subform tool (Toolbox), 446
Subform/Subreport Wizard, subforms, creating, 499-502
subforms, 1200
configuring, 429-431
creating, 499-502
records, sorting, 439-442
subject databases, 798
submenus, 1200
subnet masks, 1200
subprocedures, 1200
subqueries, 858-859, 1200
correlated subqueries, 1172
subreports, 1200
links, troubleshooting, 568
properties, 567-568
reports, adding, 564-568
structure, 555-557
unbound reports, adding, 568-569
subtrees (Registry Editor), 684
suffixes, object base names, 1212
summary data for graphs, 709-711
summary information, obtaining for QBE, 351-358
Suppliers (unformatted) file source code, 602-603
Suppliers Import Specification dialog box, 620
support files, 28
symbolic constants
Access VBA, 995-997
intrinsic, 996-997
system-defined, 996-997
symbols (SQL), 849-850
synchronizing Briefcase replicas, 965-966
syntax, 1200
debugging functions, 295-298
system administrators, 1201
system colors, 1201
system databases, 1201
system default options
advanced options, setting, 86-88
edit/find options, setting, 84-85
general option, setting, 83-84
keyboard options, setting, 85-86
setting for Access 97, 82-88
view option, setting, 83

Oracle Ex. 1009, p. 1282
Oracle v. DeBellis
IPR2024-00221

system files
configuring workgroups, 877
file sharing, creating, 868-872
system functions, 1201
system objects, 1201
system tables, 1201
system-defined symbolic constants, 996-997
System.mdw, 28
databases, Access master file, 112

T

T-1 lines, 1201
Tab Control tool (Toolbox), 446
Tab controls
configuring, 492
creating, 23, 489-495
tab orders, 1201
tab-delimited text files, 244
Table Analyzer Wizard, 30, 813-818
normalization rules, 803
Table Datasheet toolbar, 72-74
Table Datasheet view, 161
Table Design command (View menu), 57
Table menu commands, Convert Table to Text, 276
Table Names command (View menu), 453
Table Properties window, 173
table records, executing calculations, 352-356
table view
Access 97 components, 67-69
Datasheet Formatting toolbar, 74-75
Table Datasheet toolbar, 72-74
Table Wizard, 130-134
Table Wizard dialog box, 131
Table Wizard opening dialog box, 131
Table Wizard relationships dialog box, 132
table-level validation rules, 173-175
TableDef objects, 1029
Tables
30.3 DoCmd methods grouped by task, 1068-1070
30.4 DoCmd methods and arguments, 1070-1075
C.1 Personnel Actions (field properties), 1232
C.2 Personnel Actions (default field values), 1232
C.3 Personnel Actions (validation criteria), 1232-1233
C.4 Personnel Actions (key fields), 1233
C.5 Personnel Actions (entries), 1233
tables, 1201
accessing fields, 349-351
adding
fixed value lookup lists, 337-340
indexes, 154-155
lookup fields, 332-333
appending records, 378-380
attached tables, 1166
AutoNumber field, modifying, 739-742
base tables, 1167
cascading deletions, 151-152, 388
cascading updates, 151-152, 388
Categories, 52-53
conversion strategy, spreadsheet data, 735-737
converting Access 2.0, 1159
creating
in other databases, 861-862
Jet DDL, 846
make-table queries, 371-378
not-equal theta joins, 344-346
outer joins, 340-342
primary keys, 152-153
self-joins, 342-344
crosstab queries
detail levels, 365-367
fixed column headings, 363-365
custom format for data types, 125-139
Datasheet view, creating, 134-137
deleting, 160
records, 381-383
dependents, 1175
design grids
data type, 116-117
field name, 116-117
primary keys, 116-117
setting, 116-117
detail tables, 1176
display format
data types, 123-128
null values, 124-125
documents
embedding, 779-781
linking, 779-781
editing
data, 53-56
fields, 271
existing databases, adding, 137-138
exporting, 945-946
client/server databases, 945-953
external data source, verifying, 746-750
external tables
graphics, 210-212
importing, 206-209
linking, 206-209
field data types, modifying, 156-158
field properties
allow zero length, 118-119
caption, 117-118
decimal places, 117-118
format, 117-118
indexed, 118-119
input mask, 117-118
setting, 117-118
size, 117-118
validation rules, 118-119
validation text, 118-119
field sequences, rearranging, 155-156

Oracle Ex. 1009, p. 1283
Oracle v. DeBellis
IPR2024-00221

field size property, selecting, 120, 120-123
field values, default values, 145-146
fields
  adding with imported records, 252
  altering, 155
  deleting redundant, 253
  field list title bar, 349-350
  freezing, 180-181
  OLE Object data types, 122-123
  output all fields property, 350-351
  overriding properties, 509-510
  queries, 264-267
  thawing, 182-183
file formats, handling, 204-220
graphics, editing, 56-57
grids, 163
HTML, exporting, 597-600
implementing
  Design mode, 57-59
  input masks, 128-129
importing
  modifying, 251-253
  to Web pages, 655-660
index limit (Access 97), 1159
information criteria (Personnel Actions Table), 139-140
joining multitable queries, 322-323
joins
  categories, 323
  creating, 855-857
Leszynski Naming Conventions, 1208-1209
linked tables, 1185
linking, 217-218
  graphs, 723-727
  modifying, 251-253
master tables, 1186
new versus existing, 148-152
pages, 1192
Paradox, modifying, 863
Paste Errors, 249
primary keys, selecting, 152-153
printing contents, 59-62
properties
  description, 115
  filters, 115
  order by, 115
  vaildation rule, 115
  validation text, 115
records
  adding, 168-170, 175-177
  appending, 170, 433-436
  deleting, 170
  editing, 436
  exporting, 253-257
  filtering, 186-199
  finding matches, 183-185
  finding matches automatically, 186
  refreshing, 442
  replacing, 170, 250-251
  selecting, 170
  sorting, 179-183
referential integrity, enforcing, 151-152
relational databases, 796-797
relationships, 146-147
  establishing, 376-377, 750-753
  modifying, 158-159
relinking (Linked Table Manager), 218-219
restructuring, 251-252
selecting
  data, 53-56
  field data types, 119-120
  size, 119-120
setting fixed-width text fields, 122
Show Relationships command, 322-323
single column equi-joins, creating, 323-326
SQL aggregate functions, 351-352
system tables, 1201
Table Wizard, creating, 130-134
  test query, executing, 752-753
  testing
    cascading deletions, 390
    cascading updates, 390-391
    hyperlinks, 586
  text, appending, 245
  type field value, adding, 742-743
  updating multiple record values, 383-387
  validation rules, 173-175
  viewing data, 52-53
  Web pages, creating, 601-603
  Windows Clipboard
    copying, 159-160
    pasting, 159-160
  xBase, modifying, 863
tabular reports, 515
tags
  character length, 1211
  components (LNC), 1228
  creating (LNC), 1227-1228
  Data Access Objects variables, 1222-1224
  LNC procedure names, 1226
  naming conventions for VBA objects, 1220
  object names, 1211
  VBA objects variables, 1221-1224
TAPI (Telephony API), 917
TARGA cards, 700
TCP/IP (Transport Control Protocol/Internet Protocol), 914, 1201
TDI (Transport Driver Interface), 1201
templates
  Access 97 for HTML, 606-608
  queries, exporting, 608-610
Tentative Append Record button, 677
termination, looping operation, 1004
termination statements (Access VBA), 998
test queries, executing, tables, 752-753

Oracle Ex. 1009, p. 1284
Oracle v. DeBellis
IPR2024-00221

testing
cascading deletions, 390
cascading updates, 390-391
hyperlinks in tables, 586
Security Wizard, 894
validation rules, 177-178
text
appending to tables, 245
formatting in forms, 457-458
obtaining (LNC), 1229
selecting
fields, 164-165
grids, 164-165
text boxes, 164-165
text box function keys, assigning, 80
Text Box tool (Toolbox), 445
text boxes, 1201
creating, 458-463
editing keys, 163
Find What, 183
text, selecting, 164-165
Validation Rule, 171
text comparisons, Access VBA modules, 1020
text controls, editing, 425-426
text data types
databases, 795
field size property, selecting, 120-123
modifying, 158
text files, 1202
exporting, records as, 255-257
importing, 233-245
delimited files, 242-243
fixed-width text files, 244
Text Import Wizard, 234-239
advanced options, 239-241
text literals, 293
text manipulation functions, 294-295, 302-304
debugging, 303-304
string identifier characters, 303-304
text values, extracting OLE objects, 759-760
thawing fields in tables, 182-183
theta joins, 323, 1202
creating, 344-345
third normal form rule, 1202
normalization, 805-806
threads, 1202
three-tier architecture databases, 1202
thumbnail images, 679
.tif files, 699
time stamps, 1202
timers, 1202
timing events, 1062-1068
title bar, 1202
Toggle Breakpoint command (Debug menu), 1009
Toggle Button tool (Toolbox), 445
toggles, 1202
forms, 464-472
toggling, 163
toolbar, 1202
toolbars, 447-450
Access 97 Table view, 71-75
Access VBA module window, 1008-1010
buttons
changing appearance, 449
creating, 449
removing, 449
creating, 449
customizing Access 97, 75-78
Database, 52
Datasheet Formatting, 74-75
Form Design, 405, 406-408
Formatting, 405
Form Design window, 409-410
properties, modifying, 75-78
Report Print Preview mode buttons, 525
resetting (default), 449
Table Datasheet, 72-74
viewing, 77-78
Web, 30
Toolbars command (View menu), 71, 448
Toolbox, 443-451
controls, adding, 444-446
form controls, adding, 451-495
Toolbox command (View menu), 407, 444
toolbox controls
bound object frames, 569-570
check boxes, 570
command buttons, 570
reports, adding, 569-570
shapes, 570
unbound object frames, 569-570
Tools menu commands
ActiveX Controls, 1011, 1123
Add-Ins, 112, 218
Analyze, 814
Compact Database, 103
Convert, 1155
Convert Form's Macros to Visual Basic, 1082
Database Utilities, 66, 102, 162, 1155
Instant Watch, 1009
Macro, 1082, 1131
Make MDE File, 105
Merge It with MS Word, 768
Object Browser, 1009
Office Links, 201, 525, 768
Options, 81, 112, 167, 410, 872
References,112, 985, 1011, 1131
Relationships, 148, 159, 376
Replication, 957
Security, 66, 883
Spelling, 72, 407, 433, 524
Visual Basic Editor, 1131
topics, 1202
Totals command (View menu), 313
TPL (Transaction Processing Language), 838
tracking Briefcase, 956
TRANSACT-SQL, 1203
transaction processing forms, 396-397, 433-437
creating Form Wizard, 397-404

Oracle Ex. 1009, p. 1285
Oracle v. DeBellis
IPR2024-00221

TRANSACTION reserved word (ANSI SQL), 847
transactions, 1203
  user-defined transactions, 1204
TransferDatabase method, 1075
transferring data for queries, 276-278
TransferSpreadsheet method, 1075
TRANSFORM reserved word (Jet SQL), 842
translating graphical QBE expressions, 284-286
triggers, 1203
trigonometric functions, 294-295, 304-306
troubleshooting
  bound object controls, 693
  charts, columnar data, 723
  employee pictures in forms, 695-696
  Excel 97
    columns, 758
    documents, embedding, 756-758
    rows, 758
  importing, 209-210
  linking, 209-210
  query expressions, 318-319
  query parameters, 332
  referential integrity enforcement, 389
  subreport links, 568
  updatable queries, 349
  variable names, 1022
  Word 97 documents in Windows Registry, 781
Truevision, TARGA cards, 700
trust, 1203
twips, 1203
two-phase commitments, 1203
type field value, adding, 742-743
type libraries, 1203
typeface, 1203
typographic conventions for Access VBA, 980-981

U

UCM (Universal Cable Module), 957
UDT (uniform data transfer), 1204
UIs (user interfaces), 180
  improvements, 29
unbound controls (forms), 444
unbound object frames
  Excel worksheet, embedding, 755-758
  forms, adding, 701-707
  properties
    setting, 705-707
    sizing, 705-707
  toolbox controls, 569-570
Unbound Object tool (Toolbox), 446
unbound reports, 515
  adding subreports, 568-569
  multiple subreports, creating, 568-569
UNC (Uniform Naming Convention), 1203
  file sharing, 907
Undo command (Edit menu), 73, 160, 407, 433, 524, 1009
Undo command (File menu), 54
unformatted Web pages, creating, 601-603
Unfreeze All Columns command (Format menu), 183
Unhide Columns command (Format menu), 199
Unicode, 1204
UNION queries (SQL), 857-858
unique indexes, 1204
unique prefixes, 1225
unique values, selecting for multicolumn equi-joins, 329-332
UNIX, 1204
unlinked graphs
  Chart Wizard, creating, 711-715
  defined, 711-715
  properties, modifying, 714-715
Unset Database Password dialog box, 883
Update All command (Briefcase menu), 957, 970
Update My Briefcase dialog box, 970
update queries, multiple records, updating, 383-387
Update Query command (Query menu), 386, 742
updates, 1204
updating
  Briefcase replicas, 957-966
  file links, 679
  indexes, 828-830
  multiple record values in tables, 383-387
  records, query properties, 346-347
Upsizing Wizard, applications, upsizing, 947-951
URLs (Uniform Resource Locators), hyperlinks, specifying, 594-595
Use System command (Help menu), 97
User and Group Permissions dialog box, 901
user-defined data types, variables (Access VBA), 991-992
user levels, Leszynski Naming Conventions, 1209
user-defined functions, creating, 1020-1022
user-defined transactions, 1204
user-level security, file sharing, 904-906
users
  data sources, creating, 929-931
  deleting, 891-892
  managing, 884-892
  multiusers, 1188
  permissions
    granting, 895-902
    revoking, 895-902
    specific, 901-902
utilities for databases, 102-105

Oracle Ex. 1009, p. 1286
Oracle v. DeBellis
IPR2024-00221

V

validating data entry expressions, 310
validation, 1204
Validation Rule text box, 171
validation rules
data entry, 171-175
testing, 177-178
field-level validation rules, 171
table-level validation rules, 173-175
table field properties, 115-119
validation text (tables)
field properties, 118-119
properties, 115
values
extracting from OLE objects, 759-760
keys, 1184
VALUES reserved word (ANSI SQL), 847
ValuPack, 21
components, installing, 42-43
Var( ) function, 842
VARCHAR reserved word (Jet SQL), 841
variable length records, 825-826
indexes, 825-826
variables, 1204
Access VBA, 988-995
assigning, 1166
duration, 990-991
empty variables, 1178
environmental variables, 1178
explicit (Access VBA), 989
form-level variables, 1180
global variables, 1181
implicit (Access VBA), 988-989
local variables, 1185
module levels, 1187
named database objects (Access VBA), 993
names, troubleshooting, 1022
naming conventions (Access VBA), 994-995
null variables, 1189
prefixes, VBA objects, 1224-1225
scope level (Access VBA), 990-991
static variables, 1199
tags
Data Access Objects, 1222-1224
VBA objects, 1221-1224
user-defined data types (Access VBA), 991-992
variant data types, 298-300
variants, defined, 298-300
VBA (Visual Basic for Applications), 836, 1204
class modules, 1049-1050
CommandBar objects, 1075-1078
converting macros, 1080-1086
events, responding, 1060-1061
Excel 95 hierarchy, programmable objects, 1103-1104
Excel Online Help File, 1104-1106
features, 977-978
handling events, 1049
intrinsic constants, 309
macros versus code, 977-978
modifying Word 97 documents, 1118-1120
named constants, 309
objects
base names, 1219-1220
compound base names, 1220
LNC, 1209
name declarations, 1227
prefixes, combinations, 1225
tags, naming conventions, 1220
variables, 1221-1225
source codes
data event handlers, 1092-1093
event-handling code for macro conversion process, 1083-1086
event-handling code for main switchboard, 1053-1054
VBA validation subprocedure, 1091
validation subprocedure, 1091
versus macros, 1226-1227
viewing event handling code, 1050-1054
verifying tables, external data sources, 746-750
video on demand (Media Server), 579
View menu commands
Call Stack, 1009
Code, 408, 524, 982
Datasheet, 53, 716
Datasheet View, 72, 146
Debug Window, 295, 1009
Design View, 72, 146, 433
Details, 115, 969
Field List, 407, 524
Form, 407
Form Header/Footer, 406, 454
Grid, 423
Image, 696
Indexes, 58, 215
Join Properties, 341
Object Browser, 985, 1132
Page Header/Footer, 557
Pages, 525
Procedure Definition, 1011
Properties, 57, 137, 408, 524
Report Header/Footer, 557
Sorting and Grouping, 524
Source, 600
SQL, 198, 357, 753
SQL View, 843
Table Design, 57
Table Names, 453
Toolbars, 71, 448
Toolbox, 407, 444
Totals, 313
Zoom, 525
view option, setting, 83

Oracle Ex. 1009, p. 1287
Oracle v. DeBellis
IPR2024-00221

viewing
Access 97 HTML templates, 606-608
attributes of tables, 57-59
Categories table, graphics, 56-57
class modules (Object Browser), 1054-1056
data in tables, 52-53
event-handling code, 1050-1054
toolbars, 77-78
views, 1204
Datasheet, customizing, 199-201
Form toolbar, 433
forms, changing default, 464
Table Datasheet, 161
Visual Basic
Microsoft Internet technologies, 582
variant data types, 298-300
Visual Basic Editor command (Tools menu), 1131
Visual Basic for Applications 5.0, 1025-1027
Visual Basic for Applications, *see* **VBA**
Visual VBA, comparisons with Access VBA, 979-980
VM (virtual memory), 1205
VM manager, 1205
Volume table, creating, 743-746
VPN (Virtual Private Network), 1205

W

WANs (wide area networks), 1205
WAVE files, 1205
WDM (Windows Driver Model), 1205
Web
hyperlink development, 583-584
sites
ActiveX, 1096
JobForum, 635-637
Macmillan Publishing USA Information SuperLibrary, 731
World Wide Web Consortium (W3C), 600
Web pages
creating
HTML Export, 643-651
IDC, 627-637
hyperlinks, 584
queries, creating, 603-606
reports, exporting, 610-613
table data, importing, 655-660
unformatted, creating, 601-603
Web Publishing Wizard
home pages, creating, 616-618
static Web sites, creating, 614-618
Web toolbar, 30
What's This? command (Help menu), 90
WHERE keyword (SQL), 853
WHERE qualifier (SQL), 839
wild cards, 1205
Win16 function calls, converting to Win32 API, 1156
Win32 API, 1205
converting Win16 function calls, 1156
Win32 API Viewer, 1147-1150
prototype declarations (Access VBA), 1147-1150
window events, 1062-1068
Window menu commands
1 Database, 408, 434, 524
Database, 206
window objects
level one tags, 1215-1217
level two tags, 1216-1217
windows
Access 97 databases, 51-52
active, 1163
child windows, 1169
Code window, 1170
Database window, 1174
Debug window, 26, 1175
Field Properties, 172
Filter, 197
Form Design, 404-419
immediate windows, 1182
maximizing, 69-70
minimizing, 70-71
modeless windows, 1187
reports, 523-526
Table Properties, 173
Windows 95
16-bit programmable objects, interacting, 1099-1100
32-bit programmable objects, interacting, 1099-1100
automation servers, hardware requirements, 1099
networks, file sharing, 902-904
Registry, 682-689
Registry Editor, 683-686
Windows Clipboard
copying tables, 159-160
pasting tables, 159-160
Windows NT
16-bit programmable objects, interacting, 1099-1100
32-bit programmable objects, interacting, 1099-1100
NT (New Technology), 1189
Registry, 682-689
Registry Editor, 683-686
Windows NT Workstation automation servers
hardware requirements, 1099
RAM requirements, 1099
Windows Registry
manipulating automation servers, 1108-1109
troubleshooting Word 97 documents, 781
WinHelp32, 1205
Winsock, 1205
With...End structure, Access VBA object properties, 994
wizards, 30, 1164
Chart Wizard, 447
Combo Box, 475, 486-489
Create New Data Source Wizard, 230

Oracle Ex. 1009, p. 1288
Oracle v. DeBellis
IPR2024-00221

Form Wizard, 397-404, 447
Import Spreadsheet Wizard, 223
Option Group, 465-469
Partial Replica, 31
Partial Replication, 971-973
Performance Analyzer, 30
Report, 514, 516-523
Report Wizard, 447
Security Wizard, 894-895
Subform/Subreport Wizard, 499-502
Table Analyzer, 30
Table Analyzer Wizard, 813-818
Text Import Wizard, 234-245
advanced options, 240-241
Upsizing, 947-951
World Wide Web, 30
Word 97
displaying object models, 1116-1118
documents
embedding, 779-781
embedding criteria, 778-779
linking, 779-781
linking criteria, 778-779
modifying with VBA code, 1118-1120
programming in object frames, 1120-1122
linking documents with bookmarks, 588-590
Online VBA Help File, 1116-1118
sending data, 1116-1123
Windows Registry, troubleshooting documents, 781
WORK reserved word (ANSI SQL), 847
Workgroup Administrator, workgroups, choosing, 876-881
Workgroup Administrator dialog box, 871
Workgroup Owner Information dialog box, 870
workgroups, 884
choosing (Workgroup Administrator), 876-881
database permissions, 902-910
system files, configuring, 877
worksheet data (Excel 97)
embedding, 755-758
importing, 730-731
Worksheet Object command (Edit menu), 758
worksheets
linking via ODBC, 228-233
objects, closing on Excel 97, 1111-1112
workstations, 1205
Access, installing on, 867
World Wide Web Consortium (W3C) Web site, 600
World Wide Web Wizard, 30
WOSA (Windows Open Services Architecture), 917, 1206
WOW (Windows on Win32), 1206
writing
Access VBA code, programming advice, 979-980
functions for database objects, 1041-1044
Wrkgadm.exe, 28
WSIDs (workstation IDs), 934

X - Y - Z

xBase, 1206
comparisons with Access VBA, 979-980
tables, modifying, 863

Yes/No fields, 1206

Zoom command (View menu), 525
zoom option (Bound Object Frame Properties window), 696-699
zooming graphics images, 698-699

Oracle Ex. 1009, p. 1289
Oracle v. DeBellis
IPR2024-00221

QUE'S MICROSOFT® OFFICE 97 RESOURCE CENTER



For the most up-to-date information about all the Microsoft Office 97 products, visit Que's Web Resource Center at

http://www.mcp.com/que/msoffice

The web site extends the reach of this Que book by offering you a rich selection of supplementary content.

You'll find information about Que books as well as additional content about these new **Office 97 topics**:

- **Word**
- **Access**
- **Excel**
- **Outlook™**
- **PowerPoint®**
- **FrontPage™**
- **Visual Basic® for Applications**

Visit Que's web site regularly for a variety of new and updated Office 97 information.

The best resources and tips for getting things done with Office 97!

Oracle Ex. 1009, p. 1290
Oracle v. DeBellis
IPR2024-00221

Broaden Your Mind And Your Business With Que

The *Special Edition Using* series remains the most-often recommended product line for computer users who want detailed reference information. With thorough explanations, troubleshooting advice, and special expert tips, these books are the perfect all-in-one resource.



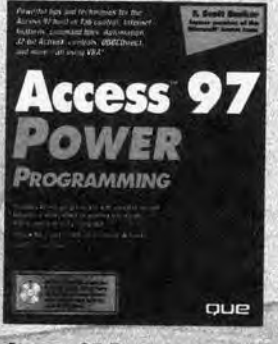
Platinum Edition Using Windows 95
- ISBN: 0-7897-0797-7-X
- $60.00 USA
- Pub Date: 4/96



Special Edition Using Windows NT Workstation 4.0
- ISBN: 0-7897-0673-3
- $49.99 USA
- Pub Date: 9/96



Special Edition Using Visual Basic for Applications
- ISBN: 0-7897-0959-7
- $49.99 USA
- Pub Date: 1/97



Access 97 Power Programming
- ISBN: 0-7897-0915-2
- $59.99 USA
- Pub Date: 12/96

For more information on these and other Que products, visit your local book retailer or call 1-800-772-0477.

que

Copyright 1995 © Macmillan Computer Publishing–A Simon & Schuster Company

Advertisement

Microsoft® FrontPage™ 97

Web Authoring and Management Tool with Bonus Pack

Professional Web Site Publishing Without Programming

Microsoft FrontPage 97 with Bonus Pack is the ideal way to get professional-quality Internet or intranet sites up and running fast. It offers all the best new Web technologies, plus powerful tools for all your Web creation and management tasks.



FrontPage 97 with Bonus Pack makes creating professional-quality Web sites effortless with powerful new functionality, support for the latest Web technologies, and seamless integration with Microsoft Office. It's never been easier!

Easy for everyone.

Microsoft FrontPage 97 with Bonus Pack can quickly turn you into a Webmaster. Use more than 30 built-in templates and wizards to build entire Web sites and individual pages easily. And with the WYSIWYG FrontPage Editor, there's no need to know HTML! Insert hyperlinks and add information from Microsoft Office and other sources with drag-and-drop simplicity. And manage your Web sites easily with the graphical tools in the FrontPage Explorer.

The best of the Web.

The latest Web technologies are at your fingertips. Drop WebBot™ components onto your pages to add such advanced functionality as full-text searching and forms. Customize your Web sites with JavaScript™ and Microsoft Visual Basic®, Scripting Edition, using an intuitive user interface. Or easily connect to databases or add ActiveX™ controls, Java applets™, and Netscape™ plug-ins for interactive, compelling Web pages.

Oracle Ex. 1009, p. 1292
Oracle v. DeBellis
IPR2024-00221

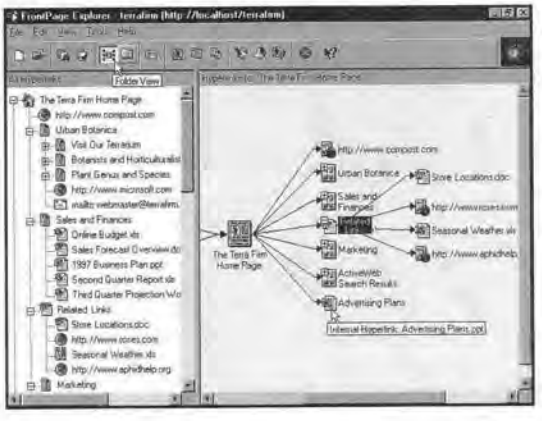
Advertisement

Professional Web site publishing is no longer just for experts!

The complete Web suite.

FrontPage 97 with Bonus Pack gives you powerful tools to create rich content and manage your Web sites effectively. Enliven your Web pages with images designed in Microsoft Image Composer, included in FrontPage 97 with Bonus Pack, or incorporate professional photographs from the Microsoft Image Composer stock photo library. Edit HTML code directly in the FrontPage Editor and preview your Web pages in any browser – without leaving FrontPage. And use advanced tools to remotely author and edit your Web sites.





Office integration.

FrontPage 97 with Bonus Pack has a familiar environment that allows you to use any document created with Microsoft Office 97 easily because it works like other Office 97 applications. Use the shared spelling checker, global Find and Replace, and the Microsoft Thesaurus to guarantee your Web sites remain accurate and compelling.

The future of Web publishing is here today! Create and manage your Web sites the fast and easy way with Microsoft FrontPage 97 with Bonus Pack.

© 1996 Microsoft Corporation. All rights reserved.
This data sheet is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. Companies, names, and/or data used in screens and sample output are fictitious, unless otherwise noted. Microsoft, ActiveX, FrontPage, Intellimouse, the Office logo, Visual Basic, Visual SourceSafe, WebBot, Windows, the Windows logo, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Netscape is a trademark of Netscape Communications Corporation. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.

Visit the Microsoft Office 97 World Wide Web site at http://www.microsoft.com/office/

Oracle Ex. 1009, p. 1293
Oracle v. DeBellis
IPR2024-00221

Complete and Return this Card for a *FREE* Computer Book Catalog

Thank you for purchasing this book! You have purchased a superior computer book written expressly for your needs. To continue to provide the kind of up-to-date, pertinent coverage you've come to expect from us, we need to hear from you. Please take a minute to complete and return this self-addressed, postage-paid form. In return, we'll send you a free catalog of all our computer books on topics ranging from word processing to programming and the internet.

Mr. ☐ Mrs. ☐ Ms. ☐ Dr. ☐

Name (first) ____ (M.I.) ____ (last) ____

Address ____

____

City ____ State ____ Zip ____ ____

Phone ____ ____ ____ Fax ____ ____ ____

Company Name ____

E-mail address ____

1. Please check at least (3) influencing factors for purchasing this book.

- Front or back cover information on book ☐
- Special approach to the content ☐
- Completeness of content ☐
- Author's reputation ☐
- Publisher's reputation ☐
- Book cover design or layout ☐
- Index or table of contents of book ☐
- Price of book ☐
- Special effects, graphics, illustrations ☐
- Other (Please specify): ____ ☐

2. How did you first learn about this book?

- Saw in Macmillan Computer Publishing catalog ☐
- Recommended by store personnel ☐
- Saw the book on bookshelf at store ☐
- Recommended by a friend ☐
- Received advertisement in the mail ☐
- Saw an advertisement in: ____ ☐
- Read book review in: ____ ☐
- Other (Please specify): ____ ☐

3. How many computer books have you purchased in the last six months?

- This book only ☐
- 2 books ☐
- 3 to 5 books ☐
- More than 5 ☐

4. Where did you purchase this book?

- Bookstore ☐
- Computer Store ☐
- Consumer Electronics Store ☐
- Department Store ☐
- Office Club ☐
- Warehouse Club ☐
- Mail Order ☐
- Direct from Publisher ☐
- Internet site ☐
- Other (Please specify): ____ ☐

5. How long have you been using a computer?

- ☐ Less than 6 months
- ☐ 6 months to a year
- ☐ 1 to 3 years
- ☐ More than 3 years

6. What is your level of experience with personal computers and with the subject of this book?

	With PCs	With subject of book
New	☐	☐
Casual	☐	☐
Accomplished	☐	☐
Expert	☐	☐

Source Code ISBN: 0-7897-0961-9

7. Which of the following best describes your job title?

- Administrative Assistant ☐
- Coordinator ☐
- Manager/Supervisor ☐
- Director ☐
- Vice President ☐
- President/CEO/COO ☐
- Lawyer/Doctor/Medical Professional ☐
- Teacher/Educator/Trainer ☐
- Engineer/Technician ☐
- Consultant ☐
- Not employed/Student/Retired ☐
- Other (Please specify): _____ ☐

8. Which of the following best describes the area of the company your job title falls under?

- Accounting ☐
- Engineering ☐
- Manufacturing ☐
- Operations ☐
- Marketing ☐
- Sales ☐
- Other (Please specify): _____ ☐

9. What is your age?

- Under 20 ☐
- 21-29 ☐
- 30-39 ☐
- 40-49 ☐
- 50-59 ☐
- 60-over ☐

10. Are you:

- Male ☐
- Female ☐

11. Which computer publications do you read regularly? (Please list)

_____
_____
_____
_____
_____
_____
_____
_____
_____

Comments: _____
_____
_____

Fold here and scotch-tape to mail.

ATTN MARKETING
MACMILLAN COMPUTER PUBLISHING
MACMILLAN PUBLISHING USA
201 W 103RD ST
INDIANAPOLIS IN 46290-9042

POSTAGE WILL BE PAID BY THE ADDRESSEE

FIRST-CLASS MAIL PERMIT NO. 9918 INDIANAPOLIS IN

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

Special Edition

USING MICROSOFT® ACCESS 97



The secrets to unleashing the full power of Access 97—at your fingertips!

Que's *Special Edition Using Microsoft Access 97* teaches you how to master the new features of 32-bit Access 97 and the Jet 3.5 database engine. From step-by-step instructions for creating relational tables to programming with VBA 5.0 and the Jet 3.5 Data Access Object, you'll quickly learn how to achieve professional results with this powerful database development platform.

The author guides you through the fundamentals of working with Access tables and queries, then leads you into the details of form and report design, writing Jet SQL, and Automation programming with VBA 5.0. You'll discover how to publish your own databases on the Web, find expert advice for replicating your databases, use Access across a network with multiuser databases, and learn how to create high-performance client/server applications using ODBCDirect. You'll also explore VBA 5.0 event-handling procedures and use ActiveX controls to customize your Access applications.

- Design and manage Access 97 databases
- Create hypertext links with HTML 3.0
- Publish databases on your Web server
- Create dynamic Web pages with user-defined queries
- Summarize and graph data with crosstab queries
- Use 32-bit OLE Automation and ActiveX™ controls
- Understand Access security features
- Create front-ends for SQL Server™ 6.5 using ODBCDirect

Raise your Access 97 productivity to new heights with Special Edition Using Microsoft Access 97 *from Que!*

User Level

New Casual Accomplished Expert

Category: Databases

Covers: Microsoft Access 97 for Windows® 95 and Windows NT® 3.51 and higher

100% Satisfaction Guaranteed

If you are not completely satisfied, you can return this product for a full refund
Call 1-800-858-7674

$49.99 USA/$70.95 CAN/45.99 Net UK

que®

Visit us on the Internet at: http://www.mcp.com/que

0 29236 09163 7

ISBN 0-7897-0916-3

94999

9 780789 709165

Oracle Ex. 1009, p. 1296
Oracle v. DeBellis
IPR2024-00221