TITLE: Microsoft Access Tips for dBASE Developers

Summary:

This article is intended to give the developer who is familiar with dBASE but new to Access information and guidance. Some of the key differences between dBASE development and Access development are discussed here. Once the dBASE developer recognizes these differences, application development in Access will prove to be stimulating and rewarding.

More Information:

Part 1
==========

Terminology

The following explanations will define some of the differences and similarities between dBASE and Access terminology.

Database

The first look of Access will seem familiar to some dBASE IV users, in that the Access database container is somewhat similar to the dBASE IV Control Center. A dBASE Catalog is similar to an Access Database and a dBASE Database is similar to an Access Table.

A dBASE IV Catalog is actually a dBASE IV database that contains information such as the names and location of databases, reports, forms, etc. that are associated with a specific Catalog. In Access, a Database contains all of the "objects" or forms, reports, etc. which are associated with the Database.

Structure
---
dBASE IV stores forms, labels, reports, etc. in separate files. In Access, all objects in a database are stored in one .MDB database file. Everything within the database file is referred to as an "object"; a form, a report, a table, are all objects.

In dBASE, the terms "field" and "record" are used. In Access, a field is sometimes referred to as a "column" and a record is occasionally called a "row". Since both terms are used interchangeably, neither is considered more correct. The Access documentation uses the terms "row" and "column" when referring to "physical entities" in a spreadsheet-like or grid-like structure, and "record" and "field" when discussing data manipulation and design.

Queries and Relationships

Relationships in Access are similar to the SET RELATION TO command or QBE in dBASE IV. Access however, allows many simultaneous or multiple relationships.

dBASE IV supports two types of queries; view queries and update queries. A view query in dBASE IV is a partial picture of one or more database files. An update query in dBASE IV allows for broad changes to be made to a single database.

A view query in Access is called a Select Query. Select Queries are used to view data in selected tables. The set of records, or the outcome of any query, is called a "dynaset". The dynaset of a Select Query can be updated under most circumstances. An Action Query in Access is used to make broad changes to data in tables. Access also supports Crosstab Queries for viewing data in a compact, spreadsheet format.

Naming Conventions

The DOS eight-character limit naming convention is used throughout dBASE because all files are stored separately. For example, a dBASE IV report has three separate files; .FRG, .FRO, and .FRM.

Access object names may be up to 64 characters and may include special characters and spaces with some exceptions: names can not begin with leading spaces, the special characters . (period), ! (exclamation point), and [ ] (square brackets) can not be used, also the control characters (ASCII 0 through 32) cannot be used. Since Access stores all objects; tables, forms, reports, etc., in one .MDB file the names are not restricted by the 8 character DOS limitation..

Field names in dBASE are limited to 10 characters; Access allows a field name to be up to 64 characters and may include spaces and special characters. The same object naming exceptions apply to field names as well.

Indexes

Handling indexes in Access involve much less work than in dBASE. Indexes are defined within the table for whatever columns are needed. From that point on, Access automatically opens all indexes and maintains them whenever a table is used. Further, indexes in Access can be set to allow or disallow duplicates during data entry.

In Access, forms use indexes to quickly reach the first record in a table/query. It is best not to use indexes in reports since Access will optimize the retrieval of data. Indexes are not used with sorting and grouping in reports unless it is set to match the multiple column index.

Field Data Types

The following table describes dBASE and Access Field Data Types. The most noticeable difference is in the way Access stores numeric data. Access also supports a binary field type allowing OLE objects to be stored giving flexibility to the database application.

dBASE	Maximum Value	Access	Maximum Value
Character	254 chars	Text	255 chars
N/A		Byte	255
N/A		Integer	-32,768 to 32,767
N/A		Long Integer	-2,147,483,648 to 2,147,483,647
Numeric	20 digits, fixed	Single	-3.402823E38 to 3.402823E38 for positive values
Float	20 digits, floating point	Double	-1.79769313486232E308 to 1.79769313486232E308
N/A		Currency	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Date	Stored internally as yyyymmdd	Date/Time	Jan. 1, 100 to Dec.31, 9999
Memo	64K chars	Memo	max. 32K chars
Logical	Y,y,N,n,T,t,F,f	Yes/No	Yes/No, True/False, On/Off
N/A		Counter	Unique long integer sequentially assigned by Access
N/A		OLE Object	

Part 2

==========

Role of the Operating Environment

The most obvious difference between dBASE and Access is the operating environment. When developing applications in dBASE, considerations must be made for printing, networking, memory limitations, video display, and more. Access runs in the Windows environment and all hardware, memory, networking, and other environmental settings are managed by Windows.

Control Panel Settings

In dBASE, a routine may be needed to check for a monochrome or color video display, then the routine would adjust the color scheme for the appropriate monitor. Once Windows is installed with the appropriate video driver, the user simply selects their own color scheme in the Windows Control Panel.

The Control Panel is also where printers are selected, fonts are installed, networks are configured, international settings are made, and sound card drivers are installed. The settings within the Control Panel influence all Windows applications. For Access, this means programs do not need to be written to configure or customize any of these environmental settings.

Advanced programmers may change the Control Panel settings by accessing Windows API using Access Basic. For more information on API calls, refer to the Microsoft Windows Software Development Kit.

Printing and Fonts

In dBASE IV, a specific printer needs to be selected and occasionally, printer control codes need to be sent. A flexible dBASE IV application would be written so the user could select a particular printer, port, etc. The dBASE developer must also take into consideration the status of the printer before sending output to the printer, whereas in Access, all of this is handled by Windows. In dBASE IV, fonts are difficult to setup and handle within procedures and desirable printouts are not easily achieved. Access on the other hand, makes use of the Windows fonts without programming to produce true WYSIWYG.

Memory Management

In dBASE IV, memory considerations are constant in application development. Screens, windows, popups, variables, etc. must be used sparingly in low memory situations and released as soon as the procedure is completed. These type of clean-up routines are not necessary in Access because memory management, including system resources, is handled exclusively by Windows. .

Help System

Help can be achieved with little programming in dBASE IV, but for more sophisticated help, such as Context Sensitive Help, an extensive routine must be written. Access developers can incorporate Help by using the Windows Help Compiler to provide Context Sensitive Help that meets the Windows

Standard. Refer to the Windows Help Compiler for more information.

User Interface

The look and feel of Access applications will be familiar to users since the Windows standard is incorporated into Access. Applications developed in Access inherently have the graphical power and consistency of other Windows applications. The Access developer has easy access to familiar Windows user-interface styles such as; buttons, combo boxes, radio buttons, etc.

dBASE IV developers use popups to give the user a list of selections, in Access, the combo boxes and list boxes provide similar but more sophisticated functionality than the dBASE equivalents. Other enhancements to the Access application development process include the flexibility of the mouse, scroll bars, check boxes, dialog boxes, and more. For more information on developing applications that meet the Windows standard, refer to the Microsoft Windows User Interface Style Guide.

Part 3
==========

Interactive Development

One of the major areas in which dBASE and Access differ is in the application development process. The Access development model is truly interactive, whereas dBASE IV provides an interactive front-end and some automated programming capability to what is essentially a traditional programming language.

Everything in dBASE IV ultimately resolves into dBASE IV code. The tools for creating forms and reports in dBASE are "generators" -- they write code. In Access, on the other hand, forms and reports exist in the database in the same way that tables do, and can be manipulated directly. Once the form or report is composed, the Access developer defines or modifies properties and uses macros to control the way these objects behave instead of "tweaking" the code.

For example, to create a form in dBASE IV, the developer might begin with the forms generator which generates dBASE IV source code. The developer would then modify the code adding additional functionality for data validation, formatting, etc. The Access developer instead embeds these instructions directly into the Form Object by way of form properties, control properties, and macros.

At the simplest level, for example, the dBASE programmer controls the order in which fields are entered on a screen by rearranging the @...SAY...GETs in the source code. The Access developer can change the order by selecting the menu option Tab Order when in form design. Access' powerful control mechanisms also allow the developer to simply move focus to a specific field based on any criteria, including the contents of another field, either on the form or in another table altogether.

Macros or Modules?

Macros in dBASE and in most other applications, are stored keystrokes used to automate repetitive

tasks. In Access, Macros are very powerful and can be used to automate procedures and perform calculations. Macros can be used to control actions on buttons, hide or display controls, position windows, and even validate data.

Most developers will find Macros to be so powerful that they do not have to write a single line of code. For dBASE developers, this may cause some re-thinking of the development process. Many typical situations where coding would be necessary in dBASE, such as multi-table forms, and complex reports, are easily created by using the rich control and function sets in Macros.

Because of the inherent power in Macros, the developer may find the line between when to use Macros and when to code somewhat unclear. As a general rule, Macros are best used for automating user-interface related tasks, whereas Modules may be needed for more complex or custom arithmetic functions. Modules also provide the means to implement complex logical constructs, such as Do..Loops and complex record manipulation, whereas Macros provide for simple control of flow.

Setting Properties

Many actions which are included in dBASE IV source code are performed in Access by setting form and report properties. For example, the field edit options in dBASE IV (such as DEFAULT, WHEN and REQUIRED) are generally implemented in Access using form control properties.

Most properties allow the developer to choose from a list of valid options. Some, however, allow the specification of a macro or expression. The following table describes the properties for which macros or expressions are specified:

Event Properties
==========

Property Name	Property of	Affects	Setting
AfterUpdate	Forms/Controls	Record/Field	Macro/Expression

Use When: After data in a field or record is updated in the database.
Example Usage: Control user interaction (move to specific control based on value of field).

Property Name	Property of	Affects	Setting
BeforeUpdate	Forms/Controls	Record/Field	Macro/Expression

Use When: Before data in a field or record is updated in the database.
Example Usage: To validate several conditions or multiple expressions.

Property Name	Property of	Affects	Setting
OnClose	Forms/Reports	Record	Macro/Expression

Use When: Before a form or report is closed.
Example Usage: Set up environment.

OnCurrent Forms Record Macro/Expression

Use When: When focus moves from one record to another.
Example Usage: Synchronize records related to the current record/Set up form environment.

OnDblClick Form/Controls Field Macro/Expression

Use When: When user double clicks control or label.
Example Usage: Overriding default control or label behavior.

OnDelete Forms Record Macro/Expression

Use When: When user deletes record.
Example Usage: Check dependencies.

OnEnter Form/Controls Field Macro/Expression

Use When: When a control gets focus.
Example Usage: 1. Set up editing environment
2. Conditional entry.

OnExit Form/Controls Field Macro/Expression

Use When: When a control loses focus, even if no changes are made in the field.
Example Usage: 1. Change tab order.
2. Complex output formatting.

OnFormat Report/Sections Record Macro/Expression

Use When: Before a report section is formatted for previewing or printing.
Example Usage: Change page layout based on data in the current record.

OnInsert Forms Record Macro/Expression

Use When: When the user types the first character in a new record.
Example Usage: Set up editing environment.

OnOpen Forms/Reports Record Macro/Expression

Use When: Before a form or report is opened.
Example Usage: Set up environment.

OnPrint Report/Sections Record Macro/Expression

Use When: Before a section is printed or previewed.
Example Usage: Changes that do not effect page layout.

OnPush Command Button Control N/A Macro/Expression

Use When: When the user presses the button.
Example Usage: Automate repetitive, but optional, actions.

ValidationRule Fields/Form controls Record/Field Expression

Use When: Evaluated when data in a field is added or changed.
Example Usage: Enforcing data integrity.

Value Properties
==========
Property Name Property of Affects Setting
==========
ControlSource Controls Field Field/Expression

Use When: Determines what is displayed in the control.
Example Usage: 1. Data which is dependent upon the value of other
controls on form.
2. Calculated data.

DefaultValue Fields/Controls Field Text/Expression

Use When: Sets the default value for a field or control.
Example Usage: 1. Minimize repetitive data entry.
2. Enforce data integrity.

Part 4
==========

PROGRAMMING TERMINOLOGY

Method

The dBASE language provides commands and functions that can be used to manipulate data within databases. Access provides Methods that act upon an object, such as a form or control. A Method is similar to a dBASE command, although it always acts upon an object.

Virtual Table

A Virtual Table, or VT, is used in Access to manipulate data. A Virtual Table is a temporary table that exists in memory and is created from a base table.

Object

Object refers to an entity such a table, form, query, etc. within a database.

Property

A Property is a named attribute of a form or control that is used to define the characteristics of an object. For example, Width is a property of a form, and Text Color is a property of a button.

Control

A Control is a graphical item on a form or report which can be used to display data, perform an action, or decorate the form or report. Some examples of Controls are lines, boxes, buttons, and charts.

PROGRAMMING TECHNIQUES

Access Basic is an event driven language, unlike a traditional hierarchical language such as dBASE. In dBASE a top-level routine calls lower-level routines, which in turn call still lower-level routines. In dBASE when a response from the user is needed, the whole program pauses in a loop while it waits for the user.

Access Basic handles user actions and displays documents and data automatically. When a user selects a menu command or a command button, the attached procedure is called. Access Basic never pauses to demand a menu choice from the user. Instead, users can ignore the menu bar completely until they need it.

The dBASE and Access languages share many similarities, since dBASE in many ways is an extension and variation of an early form of BASIC. Many dBASE statements and functions have the same or nearly the same name and perform the same operation in Access. The dBASE and Access languages are not identical, so dBASE code can not simply be run in Access Basic. Nevertheless, the similarities between the two make Access Basic an easy language for dBASE developers to learn. And, as you will see, Access Basic also provides a variety of powerful features that have no parallel in the dBASE world.

Modules

Modules in Access are very much like procedure files in dBASE. Unlike dBASE, Access Basic does not have restrictions on the number of procedures that can be stored in each module or the number of modules that can be contained in a single database.

In dBASE IV, a program is stored in a .PRG file which is then compiled into a .DBO file and can be run from the dot prompt by entering DO ProgName. Procedures, functions, and variables within dBASE programs are only available when the program is open and in memory. dBASE IV does allow a library procedure to be opened automatically making all of its procedures, variables, and functions available.

The Access approach is similar to dBASE IV libraries whereas Access opens all modules into memory once a Database is opened. Therefore, functions, procedures, and global variables are available at anytime while the database is open.

Immediate Window

The immediate window is similar to the Dot Prompt, whereas it provides a means to enter single lines of code and see the results. This technique is invaluable when debugging programs.

Declarations Section

The Declarations Section is the first section in a module. Any variable declared in this section becomes global to the module in which it exists. If the Global statement is used to declare a variable, it becomes global to other modules and objects while the database is open. Other declarations can be made in the Declaration Section such as user-defined data types, and global constants.

Procedures

In dBASE IV, procedures are created in a program by entering: PROCEDURE ProcName. Developers can specify that parameters will be passed to the procedure by using the Parameters statement. The procedure is then called by DO ProcName WITH Param1, Param2.

In Access, procedures are created within a module by entering: Sub PROCNAME () Access allows parameters to be passed to Sub procedures and Functions. To create a procedure with parameters enter:

```
Sub PROCNAME (Parameter_1 as type, Paramater_2 as type)
```

Procedures can be declared as Static so any local variables within the procedure are preserved between calls. A procedure can also be declared as Private indicating the procedure is accessible to other procedures within the module it exists but not accessible to other modules. Access Basic defines true local variables, allowing for recursive subroutines. As in the dBASE language, Access procedures do not return values, they perform operations.

Functions

A Function in Access is similar to a function in dBASE, it is a procedure that always returns a value and can be used within expressions. In Access, a function can be declared as Static meaning the variables within the function are preserved between the function calls. A function can also be declared Private which only allows the function to be accessible within the module in which it exists.

A typical dBASE function is as follows with the Return command returning the value:

```
FUNCTION FuncName(Para1, Para2)
  PARAMETERS Para1, Para2
  mVar = ...
```

```
  ...
  RETURN(mVar)
```

In Access, a function value is returned by setting the Function Name equal to the value, for example:

```
Function FUNCNAME(Para1 As Double, Para2 As String)
  Dim mVar As String
  ...
  FUNCNAME = mVar
End Function
```

Scoped Variables

The dBASE language supports PRIVATE and PUBLIC variables which are similar to the Local and Global variables of Access. Global variables that will be used within the module are created in the Declarations section of a module by entering: Dim mVar As [Type]. In dBASE, the command PUBLIC mVar is placed at the beginning of a procedure to create public, or global, variables. Access also allows variables to be defined as Global in the Declarations section which allows variables to be accessed from other modules within the database:

```
Global mVar As [Type].
```

dBASE private variables are created by issuing either mVar = 0 or STORE 0 TO mVar within a procedure or function. In Access, a variable is private, or local, if it is declared within a sub procedure or function using Dim mVar As [Type]. In Access, local variables are truly local. Unlike dBASE private variables, they are not available to procedures called by the procedure in which they are defined. In order to access a variable in this way, it must be declared in the Declarations Section with the Dim statement.

In Access, variables are initialized at compile time; numeric variables are set to 0, string variables are set to Empty.

Variable Types

In Access variable types can be either Integer, Long, Single, Double, Currency, String, or Variant. Variant is the default data type if a variable is not explicitly defined, meaning it can contain any data type. The first assignment to a Variant type variable determines the data type of the variable. There is not an actual Date type since Access stores dates as serial numbers, declare a variable as Double to store date information.

Access Basic is a strongly-typed language. Therefore, once a variable is declared and a type is assigned, the variable cannot change type as dBASE allows. In dBASE, the following is allowed:

```
mVar = 3
mVar = "Hello"
```

This is generally considered to be bad programming practice because it often hides program bugs. Access Basic will not allow incompatible value types to be assigned to variables. This is helpful when debugging programs since Access Basic will quickly reveal any incorrect assignments.

"On the Fly" Variables

In Access, variables can be created without prior declarations by using the variable type symbol. For example,

```
mVar$ = "Hello"
```

creates a string variable and

```
mVar% = 32
```

creates an integer variable. Variables that are created "on the fly", or implicitly, are considered local variables.

Arrays

In dBASE IV, arrays are declared with the DECLARE statement, Access initializes arrays similar to other variables, with the Dim statement. Access additionally allows for an Option Base which can be used to specify the beginning array column number. A dBASE IV private array is declared as:

```
DECLARE Customer[15]
```

whereas a public, or global array is declared as:

```
PUBLIC ARRAY Customer[15]
```

In Access, an array is either Local or Global depending upon where it is initialized. Just like variables, if the array is initialized in the Declaration section with the Dim statement it is Global to that module. If the array is initialized within a sub procedure or function, it becomes local. To declare a Global array that is accessible to all modules, in the Declaration section enter:

```
Global Customer(0 to 15)
```

The following examples are equivalent:

```
dBASE:  DECLARE ARRAY Customer[15]
Access: Dim Customer(1 to 16) or Dim Customer(15)
```

Arrays can be created "on the fly" within a procedure or function. Implicit arrays are always created as single dimensional with ten elements and a zero base. The following command creates an implicit array with the fourth element containing the name "Smith":

```
    Customer(3) = "Smith"
```

Notes

In dBASE IV, an * (asterisk) symbol is used to remark an entire line from a program. Use the ' (apostrophe) symbol to accomplish this in Access. The ' (apostrophe) can also be used to remark the last part of a line whereas in dBASE IV the && (double ampersand) is used.

Part 5
==========

PROGRAMMING EXAMPLES

Constructs

Looping constructs such as the Do While loop are similar between dBASE and Access. Access also supports the Do Until...Loop and a For...Next construct.

The following syntax examples show the similarities and differences between the dBASE and Access If loops:

```
dBASE                   Access
=============================================================
Example 1
--------------
IF <condition>          If <condition> Then <this> Else <this>
   <this>
ELSE
   <this>
ENDIF


Example 2
------------
IF <condition>           If <condition>  Then
  <this>                 <this>
ELSE                    Else
  <this>                 <this>
ENDIF                   End If


Example 3
```

```
-------------
IF <condition>           If <condition> Then
  <this>                   <this>
ELSE                     ElseIf <condition> Then
  IF <condition>             <this>
    <this>               Else
  ELSE                       <this>
    <this>               End If
  ENDIF
ENDIF
```

The first two examples are identical, although in Access the 'If' statement can be placed on a single line so the 'End If' is not needed. The second example is similar to the dBASE syntax and is more readable for long expressions. The final example displays the richness of the Access If loop, in dBASE the If statements are embedded, whereas Access provides this functionality with the ElseIf clause.

Access also supports the Immediate If function in the same manner as dBASE IV:

```
IIF(<expression>, <then true>, <then false>)
```

A dBASE Do Case...EndCase allows for various expressions to be evaluated which in reality is similar to Access' If... Then... ElseIf.. Then.. Else... End If construct. Access Basic's Select Case <expr>...End Select only accepts a single expression and the Case clause evaluates the result.

Transaction processing is handled similarly between dBASE and Access with Access Basic's BeginTrans... Rollback and CommitTrans statements.

Opening a Table

In dBASE IV, every database is stored as a .DBF file. In the dBASE IV command language, the following command is used to open a Customer database file:

```
USE Customer
```

To open a table in Access Basic the following methods are used:

```
Dim MyDB As Database
Dim MyTable As Table

Set MyDB = CurrentDB()
Set MyTable = MyDB.OpenTable("Customer")
```

In the above example, the first two lines are entered either in the Declarations section of a module to make the variables Public to the module, or within a sub procedure to make them local to the program. The following two lines can be entered into the immediate window or created as a function or procedure.

The first two lines declare the variables which are used to represent the database and the table. The third line initializes the variable MyDB to represent the current open database. The last line initializes the variable MyTable to represent the Customer table. The method on the last line uses the database variable name MyDB with the OpenTable method to actually open the Customer table. The variable MyTable is referred to from that point on to use the Customer table.

The following dBASE IV code shows a similar concept where the variable MyData is then used to refer to the Customer database:

```
PUBLIC MyData
MyData = "CUSTOMER"
USE &MyData
```

Opening Databases
---
dBASE IV 1.5 allows for 40 multiple databases, or tables, to be opened at once by placing the databases in separate work areas. For example, the following code would open three databases:

```
SELECT 1
USE Customer
SELECT 2
USE Orders
SELECT 3
USE Products
```

When a database is opened in Access, all of the tables within the database can be opened. Access Basic also allows for multiple Databases to be opened at once, the following is an example of this.

```
Dim MyDB as Database, MyOldDB as Database
Dim MyTable1 as Table, MyTable2 as Table, MyTable3 as Table
Dim MyOldTable1 as Table, MyOldTable2 as Table
Dim MyOldTable3 as Table

Set MyDB = CurrentDB()
Set MyOldDB = OpenDatabase("ARCHIVE")
Set MyTable1 = MyDB.OpenTable("CUSTOMER")
Set MyTable2 = MyDB.OpenTable("ORDERS")
Set MyTable3 = MyDB.OpenTable("PRODUCTS")
Set MyOldTable1 = MyOldDB.OpenTable("CUSTOMER")
Set MyOldTable2 = MyOldDB.OpenTable("ORDERS")
Set MyOldTable3 = MyOldDB.OpenTable("PRODUCTS")
```

In the dBASE code example above, the databases are referred to by their work area, for example, Select 2 calls the ORDERS database. The Access method is simply storing the Table name to a variable, so in Access the ORDERS table is referred to as MyTable2.

The following dBASE IV code allows the database to be referred to with variables which can then be opened by using SELECT MyTable2:

```
USE Customer IN 1 ALIAS MyTable1
USE Orders IN 2 ALIAS MyTable2
USE Products IN 3 ALIAS MyTable3
```

Record Movement

When using an indexed database in dBASE, there are several commands that can be used for moving the record pointer and finding data. The following chart will display the dBASE command with the Access equivalent method.

dBASE Command	Access Method
GO TOP	MoveFirst
GO BOTTOM	Movelast
GO <n>	GoToBookMark
SKIP	MoveNext
SKIP -1	MovePrevious
EOF()	EOF

dBASE Command	Access Property
SEEK	Seek
FOUND()	NoMatch

When programming in dBASE, the record number is commonly used to refer to a particular position, or record, within a database. These record numbers are stored internally and are defined by dBASE not by the programmer. Access does not use record numbers although if this programming style is preferred, it is possible to use the Counter Field Type to produce a similar, unique, number for each record.

The preferred Access Basic method is to locate a record by use of any of the Methods in the previous table, then setting a BookMark and assigning the value to a variable. After a BookMark is set, the GoToBookMark is used to return to the particular record.

The following examples display both the dBASE and Access approach to locating a specific record (assuming the databases and tables are open and variables declared, as explained earlier) and then using a bookmark and record marker to go to the previously located record:

```
dBASE                          Access
-------------------------------------------------------------
GO TOP                         MyTable.MoveFirst
DO WHILE .NOT. EOF()           Do Until Mytable.EOF
   IF LastName = "Donaldson"      If MyTable.LastName "Donaldson" Then
```

```
    mRec = RECNO()              MyMark$=MyTable.Bookmark
    EXIT                        Exit Do
  ELSE                        Else
    SKIP                        MyTable.MoveNext
  ENDIF                       End If
ENDDO                       Loop
GO mRec                     MyTable.GoToBookmark MyMark
```

The next examples will display the dBASE and Access approach to performing an indexed search.

```
dBASE                       Access
-----------------------------------------------------------------
SET ORDER TO LASTNAME         MyTable.Index = "Lastname"
SET NEAR ON                   MyTable.Seek "<=", "SM"
SEEK "SM"                   If MyTable.NoMatch Then
IF .NOT. FOUND()              Print "No matches found"
   ? "NO MATCHES FOUND"       End If
ENDIF
```

In dBASE, the SET NEAR command will control whether a record has to match the Seek expression precisely or not. Access handles this functionality by using an argument with the Seek method, the following are valid arguments:

```
Available Arguments       Meaning
-----------------------------------------------------
"="      "EQ"             Equal to
"<"      "LT"             Less than
"<="     "LE"             Less than or Equal to
">"      "GT"             Greater than
">="     "GE"             Greater than or Equal to
```

Part 6

==========

Rewriting dBASE Applications for Access

As a developer, you may be faced with the task of converting an application written in dBASE to Access, or you may have a library of procedures and techniques that you would like to use in Access. The first step in porting an application to Access is to ask yourself what you want to accomplish with the port. Which of Access' features do you want to exploit? The goals of the application need to be compared with the capabilities of Access. It is best to isolate the core processes in the old application from the user interface. A dBASE applications user interface has little in common with the graphical interface of Windows. It is much easier to discard the dBASE interface and a build a new one in Access

rather than trying to emulate the old interface. Fortunately, it is easy to use Access' tools to build forms and menus, and no programming is required to create the interface or to bring it to life.

Access supports attaching to dBASE databases which makes the Access conversion and development process easier. While developing the Access application, users can continue to work with the dBASE application and the Access developers can have access to the "live" data by attaching to the dBASE databases. Attaching to dBASE databases give users the ability to learn the Access application while maintaining the current dBASE data until the conversion process is complete.

Since dBASE data formats are recognized by Access, tables can quickly be created by importing the data. Some data types may need to be converted and columns added to take full advantage of Access' power. Forms are the core of Access. dBASE applications maintain a concept of screens which can be easily re-created in Access. The Form Wizard is an excellent tool for creating sharp forms quickly. Keep in mind that forms are generally used to gather information to be stored in the database; dialog boxes are used to gather information that the application uses to control and modify its behavior.

Converting the core data processing procedures may be the most difficult part of porting an application to Access. Many of these processes should be invoked as part of a validation constraint or event function. However, when converting to Access' event-driven structure, it is sometimes not obvious where they belong or when they should be called. When analyzing this aspect of the application, consider the original business rules that the old application was designed to enforce. Instead of trying to directly convert the code, use it as a guide to implementing those business rules in Access. Translate the "spirit" of the application, rather than the letter.

Access provides a spectrum of features that no other database application development system can match. Converting an existing application to Access is not a trivial task; however, when the power of the application can be multiplied by exploiting these features, the result is well worth the effort.