Labeling Features
Using Attributes from an External Database

By Mohammed A. Hoque, Town of West Springfield, Massachusetts

Often, spatial feature attributes are stored in an external database. For example, local governments typically store parcel location data in shapefile, geodatabase, or another GIS format while keeping ownership information in an external database maintained by the assessor's department. To print parcel maps with owners' names, the table containing the owners' names is joined with the parcel layer before labeling.

This procedure works fine as long as the relationship between the spatial and nonspatial data is one-to-one or many-to-one. However, if the relationship is one-to-many or many-to-many, the feature is labeled with only one value instead of multiple values. Parcel information for condominiums provides a good example of this type of problem. The land may be represented by a single polygon but the tax database will have multiple records for each unit on that parcel.

This limitation can easily be overcome by using the advanced labeling option in ArcMap. With the ArcMap VBScript parser, VBScript code can be written to customize labels. *[ArcMap also supports scripting with JScript.]* ActiveX Data Objects (ADO) can be used to access an external database to retrieve attributes. *[ADO is a high-level interface that provides access to data stored in a wide variety of database sources that includes not only relational databases but also nonrelational databases, folders, data files, and e-mail messages.]*

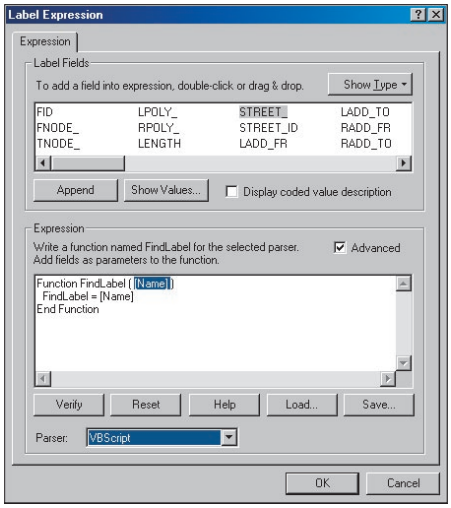
Before writing any code, two important things must be done. The appropriate database connector driver must be installed on the client PC, and the connection string for ADO must be determined. For most Windows PCs, drivers for Microsoft Access and Microsoft SQL Server are installed by default. For other databases, such as Oracle or MySQL, specific drivers can be downloaded from the software vendor's Web site. The ConnectionString is the most important ADO property. It contains the information that ADO needs to locate and configure the driver that will be used to retrieve data.

In VBScript, the CreateObject() function, with correct library and object name (e.g., ADODB.Connection), is required to create any object. Once the Connection and RecordSet the objects are created, open the database connection with the correct ConnectionString and open the RecordSet with a valid SQL statement. Retrieve the data using the Value property of the RecordSet object with the field name. Make sure that when data is retrieved, the RecordSet and Connection objects are closed and set to Nothing.

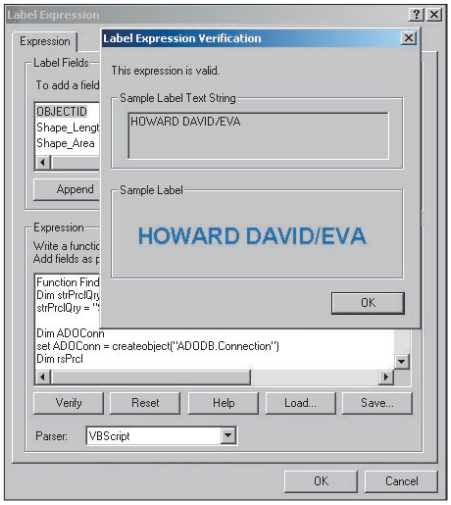
After installing the appropriate driver and finding the correct ConnectionString, follow the new steps provided to label a map layer using attribute information from an external database.

Database	ConnectionString
Oracle	"PROVIDER=OraOLEDB.Oracle; Data Source=aDatabaseName; User ID=aUserName; Password=aPassword"
MySQL	"driver={MySQL ODBC 3.51 Driver}; Server=aServerName; Database=aDatabaseName; uid=aUserName; PWD=aPassword"
Microsoft Access	"PROVIDER=Microsoft.Jet.OLEDB.4.0; Data Source=c:\myDatabase.mdb;"
Microsoft SQL Server (using Windows NT Integrated security)	"Provider=SQLOLEDB; Integrated Security=SSPI; Persist Security Info=False; Initial Catalog=aDatabaseName; Data Source=aServerName"

Figure 1: Typical connection strings for a few databases



In the Label Expression dialog box, check the Advanced box. This will write stub code for the FindLabel() function in the Expression box.



After clicking the Verify button, ArcMap will display a sample of the label if no errors are found.

1. Open ArcMap and add the layer you want to label.

2. Click on the Label tab in the Layer Properties dialog box and click the Expression button.

3. In the Label Expression dialog box, check the Advanced box. This will write stub code for the FindLabel() function in the Expression box.

4. Move the cursor between parentheses after FindLabel and double-click on the field name that will be used as the common field.

5. Write VBScript code in the Expression box similar to the code shown in Figure 2.

6. Return the string by assigning it to the function name (i.e., FindLabel = aString).

Click the Verify button to make sure the code does not contain any errors. If no errors are found, ArcMap will display a sample of the label. The Label tool from the Graphic toolbar can also be used to label features interactively.

The author has found this technique very handy because ownership information in his organization is stored in an external database, and parcel locations reside in a personal geoda-

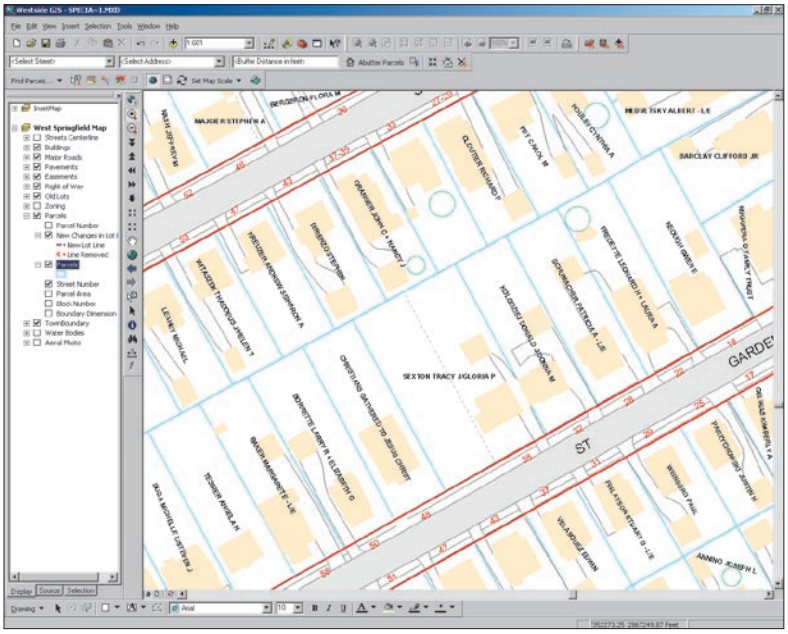
tabase that has only ID attributes.

One of the advantages of this approach is that it takes only a few seconds to label hundreds of polygons (although labeling speed will vary depending on the individual PC processor and the speed of the network used).

The only disadvantage with this method is that it creates, opens, and closes the database connection and runs the query for each visible feature on the map. This can slow down the process and generate a lot of network traffic when many features are being labeled. One way to deal with this is to set the Scale Range. On the Labels tab of the Layer Properties dialog box, click the Scale Range button and define the scale at which labels will be drawn so that fewer features will be labeled at a larger scale.

For more information, contact

Mohammed A. Hoque
GIS Coordinator
Town of West Springfield, Massachusetts
Tel.: 413-263-3070
E-mail: thoque@west-springfield.ma.us



The resulting parcel map displays the owners' names that were retrieved from an external database.

```
Function FindLabel ( [PIN] )
Dim strPrclQry, strInfo
strPrclQry = "SELECT OWNER_NAME FROM aTABLE WHERE PIN = '" & [PIN] & "'"

Dim ADOConn
set ADOConn = createobject("ADODB.Connection")
Dim rsPrcl
set rsPrcl = createObject("ADODB.Recordset")

ADOConn.Open "PROVIDER=MSDAORA;Data Source=aDatabaseService;User ID=aUserName;Password=aPassword"
ADOConn.CursorLocation = 3

rsPrcl.Open strPrclQry, ADOConn, 3, 1, 1

'If no record is found, return empty string
'if more than one records are found, put "<< More >>" to indicate multiple ownership
'you can loop through to label all owners' names
Select Case rsPrcl.RecordCount
  Case -1, 0
   strInfo = ""
  Case 1
   'reading only the first record
   strInfo = trim(rsPrcl.Fields("OWNER_NAME").Value & " ") 'so that no error will be raised in case of
NULL values
  Case Else
    strInfo = rsPrcl.Fields("OWNER_NAME").Value & "<< More >>"
End Select

'closing connections – this is a must
rsPrcl.Close
ADOConn.Close
Set rsPrcl = Nothing
Set ADOConn = Nothing

'returning string for labeling
FindLabel = strInfo
End Function
```

Figure 2: VBScript code to label parcels with owners names using ADO Connection