کار با ساختار بانک اطلاعاتی

مقدمه

یک بانک اطلاعاتی حاوی تعدادی جدول، ایندکس، قیود و چیزهای دیگری است. مجموعه این آیتمها به عنوان ساختار پایگاه داده شناخته می شود. هر دو روش DAO و ADO راههای ساده ای را برای شما مهیا کرده اند، روش مبنی بر شی برای هر دو به همان خوبی که اشیاء جدیدی در پایگاه داده می سازد، بازیابی اطلاعاتی درباره مشخصات اشیای موجود را بازیابی می کند.
ساخت یک بانک اطلاعاتی
قبل از اینکه جداول یا اشیاء دیگر را بتوانیم تعریف کنیم، باید بانک اطلاعاتی را ساخته باشیم. کد زیر یک بانک اطلاعاتی Microsoft Jet جدید را ساخته و باز می کند:

:DAO

```
Sub DAOCreateDatabase()
   Dim db As DAO.Database

   Set db = DBEngine.CreateDatabase(".\New.mdb", dbLangGeneral)
End Sub
```

:ADOX

```
Sub ADOCreateDatabase()
   Dim cat As New ADOX.Catalog

   cat.Create "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=.\New.mdb;"
End Sub
```

همانطور که ممکن است خودتان حدس زده باشید شی Database از DAO معادل شی Catalog در ADOX است. بنابراین برای ساختن یک بانک اطلاعاتی جدید از نوع Jet با استفاده از ADOX شما باید از متد Create شی Catalog استفاده کنید.
در کد DAO قبلی پارامتر Locale با dbLangGeneral مشخص شده است. در کد ADOX پارامتر Locale بصورت صریح مشخص نشده است. مقدار پیش فرض Locale برای فراهم کننده Microsoft Jet معادل dbLangGeneral است. برای تغییر مقدار Locale می توانید از خاصیت Locale Identifier استفاده کنید.
در متد CreateDatabase از DAO اغلب می توانیم پارمتر اختیاری سوم را هم در نظر بگیریم که مشخص کننده اطلاعاتی برای رمزگذاری و نسخه بانک اطلاعاتی است. برای مثال، کد زیر برای ساخت یک بانک اطلاعاتی رمز گذاری شده از نسخه Microsoft Jet 1.1 استفاده شده است.

```
Set db = DBEngine.CreateDatabase(".\New.mdb", dbLangGeneral,dbEncrypt Or dbVersion11)
```

در ADO رمز گذاری و اطلاعات نسخه بانک اطلاعاتی با استفاده از خواص مخصوص فراهم کننده مشخص می شوند. برای فراهم کننده Microsoft Jet، از خواص Engine Type(نوع موتور) و Encrypt Database (رمزگذاری بانک اطلاعاتی) استفاده می شود:

```
   cat.Create "Provider=Microsoft.Jet.OLEDB.4.0;" & "Data Source=.\New.mdb;" & _
      "Jet OLEDB:Encrypt Database=True;" & _
      "Jet OLEDB:Engine Type=2;"
```

بازیابی اطلاعات الگوی بانک اطلاعاتی

هم DAO و هم ADO هر دو دارای مجموعه ای از اشیاء هستند که می توانند برای بازیابی اطلاعات الگوی بانک اطلاعاتی بکار روند. مرور مجموعه ها، تعیین کردن ساختار اشیای بانک اطلاعاتی آسان است.

کدی که مشاهده می کنید چگونگی چاپ نام هر جدولی که در بانک اطلاعاتی قرار دارد را با استفاده از حلقه ها و مجموعه TableDefs از DAO و مجموعه Tables از ADOX را تشریح می کند.

: DAO

```
Sub DAOListTables()
  Dim db As DAO.Database
  Dim tbl As DAO.TableDef

  Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")
  For Each tbl In db.TableDefs
    Debug.Print tbl.Name
  Next
End Sub
```

: ADOX

```
Sub ADOListTables()
  Dim cat As New ADOX.Catalog
  Dim tbl As ADOX.Table

  cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=.\NorthWind.mdb;"

  For Each tbl In cat.Tables
    If tbl.Type <> "VIEW" Then Debug.Print tbl.Name
  Next
End Sub
```

شی TableDef ارائه کننده یک جدول در بانک اطلاعاتی است و مجموعه TableDefs حاوی یک شی TableDef برای هر جدول در بانک اطلاعاتی است. این کار با ADO ساده تر است، به این صورت که شی Table ارائه کننده یک جدول و مجموعه Tables حاوی همه جداول می باشد.

البته برخلاف DAO مجموعه Tables در ADO برای بانک اطلاعاتی Microsoft Access ممکن است حاوی اشیای Table باشد که جداول واقعی نباشند. برای مثال بازگشت های خطی، پرس و جوهای غیر پارامتری Microsoft Jet که در ADO با نام View ها مطرح شده اند اغلب در مجموعه Tables هستند. برای شناسایی اینکه آیا شی Table یک جدول در بانک اطلاعاتی را ارایه داده است، از خاصیت Type استفاده می کنیم. جدول زیر مقادیر ممکن برای خاصیت Type را وقتی که از ADO با فراهم کننده Microsoft Jet استفاده می کنید لیست کرده است.

نوع	توضیحات
ACCESS TABLE	شی Table یک جدول سیستم Microsoft Access است
LINK	شی Table یک جدول متصل شده از یک منبع داده غیر ODBC است
PASS-THROUGH	شی Table یک جدول متصل شده از یک منبع داده ODBC است
SYSTEM TABLE	شی Table یک جدول سیستم Microsoft Jet است
TABLE	شی Table یک جدول است
VIEW	شی Table یک بازگشت خطی، پرس و جوی غیر پارامتری است

علاوه بر اینکه می توانید اطلاعات الگوی بانک اطلاعاتی را بوسیله مجموعه های ADOX بازیابی کنید، می توانید از متد OpenSchema برای بازگرداندن یک رکوردست حاوی اطلاعاتی درباره جداول بانک اطلاعاتی استفاده کنید.

عموما استفاده از متد OpenSchema سریعتر از استفاده از مجموعه ها درون حلقه است، زیرا ADOX باید متحمل سربار ساخت اشیا برای هر عنصر در مجموعه شود. کدی که مشاهده می کنید چگونگی استفادهاز متد OpenSchema را برای چاپ همان اطلاعاتی که در مثالهای قبل با استفاده از DAO و ADOX بدست آمد را تشریح می کند:

```
Sub ADOListTables2()
  Dim cnn As New ADODB.Connection
```

```
    Dim rst As ADODB.Recordset

    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=.\NorthWind.mdb;"

    Set rst = cnn.OpenSchema(adSchemaTables)

    Do Until rst.EOF
      If rst.Fields("TABLE_TYPE") <> "VIEW" Then
        Debug.Print rst.Fields("TABLE_NAME")
      End If
      rst.MoveNext
    Loop
End Sub
```

ساختن و تغییر دادن جداول

بانکهای اطلاعاتی Microsoft Jet می توانند دارای دو نوع از جداول باشند. نوع اول جدول محلی است که در آن تعریف و داده هر دو داخل بانک اطلاعاتی ذخیره می شوند. نوع دوم جدول متصل شده است که در آن جدول در یک بانک اطلاعاتی خارجی قرار دارد اما یک اتصال به موازات یک کپی از تعریف جدول در بانک اطلاعاتی ذخیره شده است.

ساختن جداول محلی
مثالهایی که مشاهده می کنید یک جدول محلی جدید به نام Contacts می سازند.

:DAO

```
Sub DAOCreateTable()
  Dim db As DAO.Database
  Dim tbl As DAO.TableDef

  Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")

  Set tbl = db.CreateTableDef("Contacts")

  With tbl
    .Fields.Append .CreateField("ContactName", dbText)
    .Fields.Append .CreateField("ContactTitle", dbText)
    .Fields.Append .CreateField("Phone", dbText)
    .Fields.Append .CreateField("Notes", dbMemo)
    .Fields("Notes").Required = False
  End With

  db.TableDefs.Append tbl
  db.Close
End Sub
```

:ADOX

```
Sub ADOCreateTable()
  Dim cat As New ADOX.Catalog
  Dim tbl As New ADOX.Table

  cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=.\NorthWind.mdb;"

  With tbl
    .Name = "Contacts"
    .Columns.Append "ContactName", adVarWChar
    .Columns.Append "ContactTitle", adVarWChar
    .Columns.Append "Phone", adVarWChar
    .Columns.Append "Notes", adLongVarWChar
    .Columns("Notes").Attributes = adColNullable
  End With

  cat.Tables.Append tbl
  Set cat = Nothing
End Sub
```

مراحلی که برای ساخت یک جدول با استفاده از DAO و ADOX استفاده می شود شبیه هم هستند. ابتدا ساختن یک شی (TableDef یا Table)، اضافه کردن ستونها (اضافه کردن اشیای Field یا Column)، و سرانجام اضافه کردن جدول به مجموعه. اگرچه مراحل شبیه به هم هستند ولی از نظر نحوی تفاوتهای اندکی وجود دارد.

با ADOX لازم نیست که از متد Create برای ساخت ستون قبل از اضافه کردن آن به مجموعه استفاده کنیم. متد Append می تواند برای هر دو کار ساختن و اضافه کردن ستون بکار رود.

شما حتما توجه خواهید کرد که نامهای انواع داده برای ستونها در DAO و ADOX متفاوت است. جدولی که مشاهده می کنید چگونگی انواع داده DAO که می توانید برای بانکهای اطلاعاتی Microsoft Jet بکار ببری و مقایسه آنها با انواع داده در ADOX را به شما نشان می دهد:

انواع داده DAO	انواع داده ADOX
dbBinary	adBinary
dbBoolean	adBoolean
dbByte	adUnsignedTinyInt
dbCurrency	adCurrency
dbDate	adDate
dbDecimal	adNumeric
dbDouble	adDouble
dbGUID	adGUID
dbInteger	adSmallInt
dbLong	adInteger
dbLongBinary	adLongVarBinary
dbMemo	adLongVarWChar
dbSingle	adSingle
dbText	adVarWChar

تغییر یک جدول موجود

وقتی یک جدولی از قبل ساخته شده است شما ممکن است بخواهید تغییراتی را در آن مثلا برای اضافه و یا حذف کردن ستونها، قانون معتبر سازی (Validation Rule) و یا تازه کردن اطلاعات جدول متصل شده، انجام دهید.

کدهایی را که مشاهده می کنید چگونگی اضافه کردن یک ستون جدید با قابلیت افزایش خودکار به یک جدول موجود را شرح می دهند:

:DAO

```
Sub DAOCreateAutoIncrColumn()
  Dim db As DAO.Database
  Dim tbl As DAO.TableDef
  Dim fld As DAO.Field

  Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")

  Set tbl = db.TableDefs("Contacts")

  Set fld = tbl.CreateField("ContactId", dbLong)
  fld.Attributes = dbAutoIncrField

  tbl.Fields.Append fld
  db.Close
End Sub
```

:ADOX

```
Sub ADOCreateAutoIncrColumn()
  Dim cat As New ADOX.Catalog
  Dim col As New ADOX.Column

  cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=.\NorthWind.mdb;"

  With col
    .Name = "ContactId"
    .Type = adInteger
    Set .ParentCatalog = cat
    .Properties("AutoIncrement") = True
  End With
```

```
    cat.Tables("Contacts").Columns.Append col
    Set cat = Nothing
End Sub
```

در این مثال ADOX توجه کنید که خاصیت ParentCatalog از شی Column قبل از اینکه خاصیت AutoIncrement در مجموعه Properties را به True ست کنیم مقداردهی شده است. در صورت دسترسی به هر خاصیت در مجموعه Properties شی column باید به یک فراهم کننده مرتبط شده باشد.

سخت کلیدها و ارتباطهای بین جداول

قبلا ساختار یک جدول تعریف شده است، حالا بهتر است کلیدهایی برای جدول و ارتباطهایی بین جداول بسازیم. Microsoft Jet اطلاعات تهیه شده در تعریف کلید و ارتباط را برای بهینه کردن پرس و جو ها استفاده می کند.

ساختن یک کلید اصلی

یک جدول اغلب یک ستون یا ترکیبی از ستونها دارد که بر اساس آن یک ردیف در جدول منحصربفرد است. این ستون (یا ترکیبی از ستونها) کلید اصلی یک جدول نامیده می شود. وقتی شما یک کلید اصلی تعریف می کنید، موتور بانک اطلاعاتی Microsoft Jet یک ایندکس برای تاکید یکتا بودن کلید می سازد.

در این مثال از جدول Contacts که در مثال قبل ساخته شده است استفاده می کنیم، کد زیر چگونگی تعیین ستون ColumnID به عنوان کلید اصلی را شرح می دهد:

:DAO

```
Sub DAOCreatePrimaryKey()
   Dim db As DAO.Database
   Dim tbl As DAO.TableDef
   Dim idx As DAO.Index

   Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")
   Set tbl = db.TableDefs("Contacts")

   Set idx = tbl.CreateIndex("PrimaryKey")
   idx.Primary = True
   idx.Fields.Append idx.CreateField("ContactId")

   tbl.Indexes.Append idx

   db.Close
End Sub
```

:ADOX

```
Sub ADOCreatePrimaryKey()
   Dim cat As New ADOX.Catalog
   Dim tbl As ADOX.Table
   Dim pk As New ADOX.Key

   cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=.\NorthWind.mdb;"
   Set tbl = cat.Tables("Contacts")

   pk.Name = "PrimaryKey"
   pk.Type = adKeyPrimary
   pk.Columns.Append "ContactId"

   tbl.Keys.Append pk

   Set cat = Nothing
End Sub
```

با DAO شی Index برای ساختن کلید اصلی استفاده می شود. کلید اصلی مانند هر ایندکس دیگری ساخته شده است با این تفاوت که خاصیت Primary آن به مقدار True تنظیم می شود. (جادوگر: به خاطر اینکه مبحث کوتاهتر بشود قسمت ساخت ایندکس را حذف کردم) اما ADO یک شی به نام Key دارد که برای ساخت کلیدهای جدید به کار می رود. مراحل ساخت یک کلید شبیه به ساخت ایندکس است. اما وقتی یک شی Key ساخته شد شما باید نوع کلیدی را که می خواهید بسازید تعیین کنید. در اینجا نوع کلید adKeyPrimary است که نشان میدهد شما می خواهید یک کلید اصلی بسازید.

همچنین می توانید کد مربوط به ADOX برای ساخت و اضافه کردن کلید را در یک خط کد بنویسید. مثال زیر را نگاه کنید:

```
  pk.Name = "PrimaryKey"
  pk.Type = adKeyPrimary
  pk.Columns.Append "ContactId"

  tbl.Keys.Append pk
```

به جای کد بالا می توانید کد زیر را بنویسید:

```
  tbl.Keys.Append "PrimaryKey", adKeyPrimary, "ContactId"
```

ساخت ارتباطهای یک به چند (کلیدهای خارجی)

ارتباطهای یک به چند بین جداول (جایی که مقدار کلید اصلی در جدول اصلی می تواند در چندین ردیف در جدول خارجی ظاهر شود) بوسله ساختن کلیدهای خارجی برقرار می شوند. یک کلید خارجی ستون یا ترکیبی از ستونها است که مقادیرشان با کلید اصلی جدول دیگر همسان است. برخلاف کلید اصلی، یک کلید خارجی منحصربفرد نیست.

:DAO

```
Sub DAOCreateForeignKey()
  Dim db As DAO.Database
  Dim rel As DAO.Relation
  Dim fld As DAO.Field

  Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")

  db.Relations.Delete "CategoriesProducts"

  Set rel = db.CreateRelation()
  rel.Name = "CategoriesProducts"
  rel.Table = "Categories"
  rel.ForeignTable = "Products"

  Set fld = rel.CreateField("CategoryId")

  fld.ForeignName = "CategoryId"

  rel.Fields.Append fld

  db.Relations.Append rel
End Sub
```

:ADOX

```
Sub ADOCreateForeignKey()
  Dim cat As New ADOX.Catalog
  Dim tbl As ADOX.Table
  Dim fk As New ADOX.Key

  cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=.\NorthWind.mdb;"

  Set tbl = cat.Tables("Products")

  tbl.Keys.Delete "CategoriesProducts"

  fk.Name = "CategoriesProducts"
  fk.Type = adKeyForeign
  fk.RelatedTable = "Categories"

  fk.Columns.Append "CategoryId"

  fk.Columns("CategoryId").RelatedColumn = "CategoryId"
```

```
    tbl.Keys.Append fk

    Set cat = Nothing
End Sub
```

همچنین کد مربوط به ADOX برای ساختن و اضافه کردن کلید می تواند بصورت خلاصه تر در یک خط کد نوشته شود. کد زیر را نگاه کنید:

```
    fk.Name = "CategoriesProducts"
    fk.Type = adKeyForeign
    fk.RelatedTable = "Categories"

    fk.Columns.Append "CategoryId"

    fk.Columns("CategoryId").RelatedColumn = "CategoryId"

    tbl.Keys.Append fk
```

معادل آن بصورت خلاصه بدین شکل است:

```
    tbl.Keys.Append "CategoriesProducts", adKeyForeign,"CategoryId", "Categories", "CategoryId"
```

ساخت و ویرایش پرس و جوها

شی Command از ADO مشابه شی QueryDef در DAO است که برای تعیین رشته SQL و پارامترها و اجرای پرس و جو بکار می روند.

هر چند برخلاف شی QueryDef از DAO شی Command از ADO نمی تواند بصورت مستقیم برای یک پرس و جوی پایدار استفاده شود. وقتی که شی QueryDef شاخته می شود با استفاده از تعیین یک نام برای آن، شی QueryDef از DAO بصورت خودکار به مجموعه QueryDefs اضافه می شود و در بانک اطلاعاتی قرار می گیرد. تفاوت آن با شی Command در ADO در این است که همه اشایای Command بصورت پرس و جوهای موقتی هستند. شما باید بصورت صریح Command را به مجموعه های Views یا Procedures از ADOX برای پایدار کردن آن در بانک اطلاعاتی اضافه کنید.

فراهم کننده Microsoft Jet پرس و جوهای Microsoft Jet را اگر بصورت پرس و جوهای برگرداننده ردیف (row-returning) و بدون پارامتر باشند بصورت Viewها تعریف می کند. این فراهم کننده یک روال را بصورت یک پرس و جوی غیر برگرداننده ردیف (یک عمل عمده) یا یک پرس و جوی برگرداننده ردیف پارامتر دار در نظر می گیرد.

ساختن یک پرس و جوی ذخیره شده

کدهای زیر چگونگی ساختن یک پرس و جوی برگرداننده ردیف و بدون پارامتر را نشان می دهند.

:DAO

```
Sub DAOCreateQuery()
  Dim db As DAO.Database
  Dim qry As DAO.QueryDef

  Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")

  Set qry = db.CreateQueryDef("AllCategories", "SELECT * FROM Categories")
  db.Close
End Sub
```

:ADOX

```
Sub ADOCreateQuery()
  Dim cat As New ADOX.Catalog
  Dim cmd As New ADODB.Command

  cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & "Data Source=.\NorthWind.mdb;"

  cmd.CommandText = "SELECT * FROM Categories"
  cat.Views.Append "AllCategories", cmd

  Set cat = Nothing
End Sub
```

در این مثال، عبارت SQL استفاده شده بدون پارامتر و برگرداننده ردیف است، شی Command از ADO به مجموعه Views از ADOX اضافه شده است. توجه کنید که وقتی فراهم کننده Microsoft Jet استفاده می شود، می توانید شی Command را صرفنظر از نوع پرس و جویی که شروع به ساختن کرده ایم، به یکی از مجموعه های Views یا Procedures اضافه کنیم. هر چند اگر یک پرس و جو مانند این مثال را به مجموعه Procedures اضافه کنید و سپس مجموعه های Procedures و Views دوباره سازی (refresh) شوند، شما در خواهید یافت که پرس و جو در مجموعه Procedure قرار ندارد و حالا در مجموعه Views است.

همچنین شما می توانید یک پرس و جوی پارامتر دار یا یک پرس و جوی غیر بازگرداننده ردیف را به یکی از مجموعه های Procedures یا Views اضافه کنید. هرچند که ADOX در واقع این نوع از پرس و جو ها را در مجموعه Procedures ذخیره می کند. اگر شما آن را به مجموعه Views اضافه و سپس هر دو مجموعه را دوباره سازی کنید، شما در خواهید یافت که پرس و جوی اضافه شده حالا در مجموعه Procedures قرار دارد.

ساختن یک پرس و جوی پارامتر دار

کدهای زیر چگونگی ساخت یک پرس و جوی پارامتر دار و ذخیره آن در بانک اطلاعاتی را به شما نشان می دهد.

:DAO

```
Sub DAOCreateParameterizedQuery()
  Dim db As DAO.Database
  Dim qry As DAO.QueryDef

  Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")

  Set qry = db.CreateQueryDef("Employees by Region","PARAMETERS [prmRegion] TEXT(255);" & _
    "SELECT * FROM Employees WHERE Region = [prmRegion]")

  db.Close
End Sub
```

ADOX:

```
Sub ADOCreateParameterizedQuery()
  Dim cat As New ADOX.Catalog
  Dim cmd As New ADODB.Command

  cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & "Data Source=.\NorthWind.mdb;"

  cmd.CommandText = "PARAMETERS [prmRegion] TEXT(255);" & _
    "SELECT * FROM Employees WHERE Region = [prmRegion]"

  cat.Procedures.Append "Employees by Region", cmd

  Set cat = Nothing
End Sub
```

کد ساختن یک پرس و جوی پارامتر دار بسیار شبیه به استفاده از ADOX و DAO است. توجه کنید که اگرچه شی Command از ADO به شما اجازه می دهد با استفاده از متد CreateParameter پارامترها را بسازید اما این اطلاعات وقتی یک روال ساخته یا به روز شد ذخیره نشده اند. شما باید پارامترها را بصورت بخشی از رشته SQL تعریف کنید.

همچنین توجه کنید که Microsoft Jet عبارت SQL را وقتی یک پرس و جو با ADOX و فراهم کننده Microsoft Jet ساخته شده است بطور متفاوتی از DAO تفسیر خواهد کرد. فراهم کننده Microsoft Jet همیشه گزینه هایی از موتور بانک اطلاعاتی Microsoft Jet را به خصوصیات مورد قبول موسسه استانداردهای ملی آمریکا (ANSI) تنظیم می کند. این مسئله ممکن است باعث تفاوتهایی در رفتار DAO و ADO در زمان ساخته و یا اجرا شدن پرس و جوها بشود. برای مثال اگر عبارت SQL در کد قبلی بصورت زیر نوشته شده باشد:

```
  "PARAMETERS [prmRegion] TEXT;" & _
    "SELECT * FROM Employees WHERE Region = [prmRegion]"
```

عبارت (255) بعد از لغت کلیدی، تعیین می کند که وقتی که از DAO استفاده می کنید پارامتر باید یک فیلد متنی (dbText یا adVarWChar) باشد ، اما وقتی که از ADO استفاده می کنید بصورت فیلد Memo (dbMemo یا adLongVarWChar) در نظر گرفته می شود.

بعلاوه بعضی از عبارات SQL که با استفاده از DAO استفاده می شوند هنگام استفاده در ADO به دلیل کلمات رزرو شده اضافی به خطا منجر خواهند شد. برای اطلاعات بیشتر در این زمینه به لیست کلمات رزرو شده ANSI برای Microsoft Jet مراجعه کنید.

اصلاح یک پرس و جوی ذخیره شده

کدهای زیر چگونگی اصلاح یک پرس و جوی ذخیره شده را به شما نشان می دهند.

DAO

```
Sub DAOModifyQuery()
  Dim db As DAO.Database
  Dim qry As DAO.QueryDef

  Set db = DBEngine.OpenDatabase(".\NorthWind.mdb")

  Set qry = db.QueryDefs("Employees by Region")
```

```
  qry.SQL = "PARAMETERS [prmRegion] TEXT(255);" & _
    "SELECT * FROM Employees WHERE Region = [prmRegion] " & "ORDER BY City"

  db.Close
End Sub
```

:ADO

```
Sub ADOModifyQuery()
  Dim cat As New ADOX.Catalog
  Dim cmd As ADODB.Command

  cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & "Data Source=.\NorthWind.mdb;"

  Set cmd = cat.Procedures("Employees by Region").Command

  cmd.CommandText = "PARAMETERS [prmRegion] TEXT(255);" & _
    "SELECT * FROM Employees WHERE Region = [prmRegion] " & "ORDER BY City"

  Set cat.Procedures("Employees by Region").Command = cmd

  Set cat = Nothing
End Sub
```

در کد مربوط به ADO خاصیت Command شی Procedure به شی Command اصلاح شده برای ذخیره تغییرات تنظیم شده است. اگر این مرحله آخر انجام نمی شد تغییرات در بانک اطلاعاتی پایدار نمی شد. این تفاوت از این حقیقت ناشی می شود که اشیای Command از ADO بصورت پرس و جوهای موقتی تنظیم شده اند در حالی که اشیای QueryDef از DAO بصورت پرس و جوهای ذخیره شده طراحی شده اند. شما باید وقتی که با Commandها و Procedure ها و View ها کار می کنید مراقب این مسئله باشید.

منبع: MSDN
ترجمه: محمد صافدل
وبلاگ: http://vblog.persianblog.com