Comandos Shape en general

Un comando Shape define la estructura de un **Recordset** jerárquico y los comandos necesarios para poblarlo de datos.

Una parte del comando Shape es una consulta emitida al proveedor de datos subyacente que devuelve un objeto **Recordset**. La sintaxis de la consulta depende de los requisitos del proveedor de datos principal. Normalmente ésta será SQL, aunque ADO no necesita el uso de ningún lenguaje de consulta en particular.

Podría utilizar una cláusula **JOIN** de SQL para relacionar dos tablas, sin embargo, un **Recordset** jerárquico representará la información de forma más eficaz. Cada fila de un **Recordset** creado mediante un **JOIN** repite la información de forma redundante de una de las tablas. Un **Recordset** jerárquico sólo tiene un **Recordset** primario para cada uno de los objetos **Recordset** secundarios.

Puede utilizar paréntesis para agrupar partes por claridad.

Los comandos Shape sólo se pueden emitir mediante objetos **Recordset**.

Los comandos Shape pueden anidarse. Es decir, el *comando primario* o el *comando secundario* pueden ser otro comando Shape.

Para obtener más información sobre la exploración de un **Recordset** jerárquico, consulte Tener acceso a las filas de un Recordset jerárquico.

Gramática formal de Shape

Ésta es la gramática formal para crear cualquier comando Shape.

- Los términos gramaticales necesarios son cadenas de texto delimitadas por corchetes angulares ("<>").
- Los términos opcionales se delimitan mediante corchetes ("[ ]").
- Las alternativas se indican mediante una barra vertical ("|").
- Las alternativas repetitivas se indican mediante puntos suspensivos ("...").
- *Alpha* indica indica una cadena de letras alfabéticas.
- *Digit* indica una cadena de números.
- *Unicode-digit* indica una cadena de dígitos unicode.
- Todos los demás términos son literales.

TérminoDefinición

```
<shape-command>   SHAPE <table-exp> [[AS] <alias>][<shape-action>]

<table-exp>   {<native-sql-statement>} |
(<shape-command>)

<shape-action>   APPEND <aliased-field-list> |
COMPUTE <aliased-field-list>
   [BY <field-list> [[AS] <alias>]] |
BY <field-list> [[AS] <alias>]

<aliased-field-list>   <aliased-field> [, <aliased-field...]

<aliased-field>   <field-exp> [[AS] <alias>]

<field-exp>   (<relation-exp>) |
<calculated-exp>

<relation_exp>   <table-exp> [[AS] <alias>]
   RELATE <relation-cond-list>

<relation-cond-list>   <relation-cond> [, <relation-cond>...]

<relation-cond>   <field-name> TO <child-ref>

<child-ref>   <field-name> |
PARAMETER <param-ref>

<param-ref>   <number>

<field-list>   <field-name [, <field-name>]

<calculated-exp>   SUM(<qualified-field-name>) |
```

AVG(<qualified-field-name>) |
MIN(<qualified-field-name>) |
MAX(<qualified-field-name>) |
COUNT(<alias>) |
STDEV(<qualified-field-name>) |
ANY(<qualified-field-name>) |
CALC(<expresion>)

<qualified-field-name> <alias>.<field-name> |
<field-name>

<alias> <quoted-name>

<field-name> <quoted-name>

<quoted-name> "<string>" |
'<string>' |
<name>

<name> *alpha* [*alpha* | *digit* | _ | # ...]

<number> *digit* [*digit*...]

<string> *unicode-char* [*unicode-char*...]

<expression> Una expresión reconocida por el servicio de expresiones Jet cuyos operandos sean distintos a los de columnas que no sean CALC de la misma fila.

Información de jerarquía (Cuadro de diálogo)

Puede ver la información específica de la jerarquía del objeto **Command** seleccionado en el cuadro de diálogo **Información de jerarquía**. Este cuadro de diálogo sólo está disponible para el objeto **Command** primario de nivel superior. El cuadro de edición muestra la información tal como se especifique con **Ver comando Forma** o **Ver jerarquía ADO**.

Ver comando Forma

Si esta opción está activada, el control de edición muestra el comando **Shape** subyacente de la jerarquía. Por ejemplo:

```
RsCustomers
Orders
```

Ver jerarquía ADO

Si esta opción está activada, la presentación en el cuadro de edición es una lista con sangría de los objetos **RecordSet** subyacentes que crean la jerarquía. Por ejemplo:

```
SHAPE {SELECT * FROM 'customers'}  AS Customers APPEND ({SELECT * FROM
'orders'}  AS Orders RELATE CustomerID TO CustomerID) AS Orders
```

Resumen de cursores jerárquicos y formación de datos

ADO 2.0 presenta la función cursor jerárquico, que permite definir un objeto **Recordset** secundario como el valor de un campo en un **Recordset** principal.

Esta es una forma sencilla de ver los cursores jerárquicos. Imagine un control visual, como el cuadro de diálogo **Abrir archivo**, que muestra los archivos y subdirectorios de una forma jerárquica. Piense en cada directorio como un objeto **Recordset**, en cada archivo dentro de un directorio como un objeto **Field** y en cada subdirectorio dentro de un directorio como un objeto **Field** cuyo valor es otro **Recordset**.

ADO 2.0 presenta además una nueva sintaxis de lenguaje de manipulación de datos de tipo Shape, que le permite realizar consultas que den como resultado un **Recordset** jerárquico. Un comando de lenguaje Shape se emite como se haría con cualquier cadena de comando ADO.

El lenguaje Shape está incorporado en ADO Client Cursor Engine (Motor de cursores cliente de ADO) Llamaremos al proceso de crearlo *Formación de datos.*

El lenguaje Shape le permite crear objetos Recordset jerárquicos de dos formas. La primera agrega un **Recordset** de nivel inferior al **Recordset** de nivel superior y la segunda calcula una operación agregada a un **Recordset** secundario y genera un **Recordset** principal.

Puede anidar los objetos **Recordset** jerárquicos a cualquier profundidad que se requiera (esto es, crear objetos **Recordset** secundarios dentro de objetos **Recordset** secundarios, y así sucesivamente).

Puede tener acceso al **Recordset** jerárquico resultante con un programa o mediante un control visual adecuado.

El lenguaje Shape es relativamente difícil de escribir. Por ello, Microsoft proporciona una herramienta visual que genera comandos por usted. (consulte el tema de Visual Basic, "El diseñador del entorno de datos") y otra herramienta visual mostrará los cursores jerárquicos (consulte el tema de Visual Basic, "Utilizar el control jerárquico Flexgrid de Microsoft").

Recordset, objeto (ADO)

Un objeto **Recordset** representa todo el conjunto de registros de una tabla o del resultado de un comando ejecutado. En cualquier momento, el objeto **Recordset** sólo hace referencia a un único registro dentro del conjunto, llamado registro actual.

Comentarios

Los objetos **Recordset** se utilizan para manipular los datos de un proveedor. Cuando se utiliza ADO, se manipulan los datos casi completamente con objetos **Recordset**. Todos los objetos **Recordset** se construyen utilizando registros (filas) y campos (columnas). Dependiendo de la funcionalidad aceptada por el proveedor, algunos métodos o propiedades del objeto **Recordset** puede que no estén disponibles.

ADOR.Recordset y **ADODB.Recordset** son ProgID que se utilizan para crear objetos **Recordset**. Los objetos **Recordset** que resultan se comportan de forma idéntica, independientemente del ProgID. **ADOR.Recordset** se instala con Internet Explorer de Microsoft®; **ADODB.Recordset** se instala con ADO. El comportamiento de un objeto **Recordset** está afectado por su entorno (esto es, cliente, servidor, Internet Explorer, etc.). Las diferencias se describen en los temas de Ayuda de sus propiedades, métodos y eventos.

Hay cuatro tipos diferentes de cursores en ADO:

- **Cursor dinámico**: le permite ver inserciones, modificaciones y eliminaciones de otros usuarios, y permite todos los tipos de movimientos a través del **Recordset** que estén relacionados con marcadores; permite marcadores si el proveedor los acepta.

- **Cursor de conjunto de claves**: se comporta como un cursor dinámico, excepto que impide ver registros agregados por otros usuarios, e impide el acceso a registros eliminados por otros usuarios. Las modificaciones en los datos efectuadas por otros usuarios siguen siendo visibles. Acepta siempre marcadores y por lo tanto permite todos los tipos de movimientos a través del **Recordset**.

- **Cursor estático**: proporciona una copia estática de un conjunto de registros para que se utilicen en búsquedas de datos o para generar informes; permite siempre los marcadores y por lo tanto permite todos los tipos de movimientos a través del **Recordset**. Las inserciones, modificaciones o eliminaciones efectuadas por otros usuarios no serán visibles. Este es el único tipo de cursor permitido cuando se abre un objeto **Recordset** en el lado del cliente (ADOR).

- **Cursor de tipo Forward-only**: se comporta de forma idéntica al cursor dinámico excepto en que sólo le permite recorrer los registros hacia delante. Esto aumenta el rendimiento en situaciones en las que sólo tenga que efectuar un paso a través de un **Recordset**.

Establezca la propiedad **CursorType** antes de abrir el **Recordset** para elegir el tipo de cursor, o pase un argumento ***CursorType*** con el método **Open**. Algunos proveedores no aceptan todos los tipos de cursores. Compruebe la documentación del proveedor. Si no se especifica el tipo del cursor, ADO abre un cursor de tipo Forward-only de manera predeterminada.

Cuando se utilizan con algunos proveedores (como Microsoft ODBC Provider para OLE DB junto con Microsoft SQL Server), se pueden crear objetos **Recordset** independientemente de un objeto Connection definido previamente pasando una cadena de conexión al método **Open**. ADO sigue creando un objeto **Connection**, pero no asigna dicho objeto a una variable de objeto. Sin embargo, si se están abriendo varios objetos **Recordset** en la misma conexión, se tiene que crear y abrir explícitamente un objeto **Connection**; así se asigna el objeto **Connection** a una variable de objeto. Si no se utiliza dicha variable de objeto cuando se abren los objetos **Recordset**, ADO crea un nuevo objeto **Connection** por cada nuevo **Recordset**, incluso si se pasa la misma cadena de conexión.

Se pueden crear tantos objetos **Recordset** como sea necesario.

Cuando se abre un **Recordset**, el registro actual está situado en el primer registro (si lo hay) y las propiedades **BOF** y **EOF** están establecidas a **False**. Si no hay registros, los valores de las propiedades **BOF** y **EOF** son **True**.

Pueden utilizarse los métodos **MoveFirst**, **MoveLast**, **MoveNext** y **MovePrevious**, así como el método **Move**, y las propiedades **AbsolutePosition**, **AbsolutePage** y **Filter** para volver a colocar el registro actual, asumiendo que el proveedor acepta la funcionalidad necesaria. Los objetos **Recordset** de tipo Forward-only sólo aceptan el método **MoveNext**. Cuando se utilizan métodos **Move** para visitar todos los registros (o para enumerar el **Recordset**), se puede utilizar las propiedades **BOF** y **EOF** para saber si ha llegado al principio o al final del **Recordset**.

Los objetos **Recordset** pueden aceptar dos tipos de actualización: inmediata y por lotes. En la actualización inmediata, todas las modificaciones se escriben inmediatamente en el origen de datos después de invocar el método **Update**. También se pueden pasar matrices de valores como parámetros en los métodos **AddNew** y **Update** y actualizar de forma simultánea varios campos de un registro.

Si un proveedor acepta la actualización por lotes, se puede hacer que el proveedor guarde en la caché las modificaciones efectuadas en varios registros y transmitirlos después en una sola llamada a la base de datos con el método **UpdateBatch**. Esto se aplica a las modificaciones efectuadas con los métodos **AddNew**, **Update** y **Delete**. Después de invocar el método **UpdateBatch**, se puede utilizar la propiedad **Status** para comprobar si ha habido algún conflicto en los datos para resolverlo.

Nota Para ejecutar una consulta sin utilizar un objeto **Command**, pase una cadena de consulta al método **Open** de un objeto **Recordset**. Sin embargo, se requiere un objeto **Command** cuando quiera que el texto del comando persista para volver a ejecutarlo, o cuando utilice parámetros en la consulta.

Proveedores necesarios para la formación de datos

Un proveedor de origen de datos de base de datos OLE debe proporcionar datos que se pasarán a otro proveedor, que llevará a cabo la formación de datos. El proveedor de origen de datos se especifica en la cadena de conexión del objeto **Connection** como "Shape Provider=*SuOrigenDeDatos*". El proveedor que suministra el soporte de formación de datos se especifica en la propiedad **Provider** del objeto **Connection** como "MSDataShape".

Ejemplo

```
Dim cnn As New ADODB.Connection

cnn.Provider = "MSDataShape"

cnn.Open "Shape
Provider=MSDASQL;DSN=vfox;uid=sa;pwd=vfox;database=pubs”
```

Tener acceso a las filas de un Recordset jerárquico

El siguiente ejemplo muestra los pasos necesarios para tener acceso a las filas de un **Recordset** jerárquico.

1. Los objetos **Recordset** de las tablas **authors** y **titleauthors** están relacionados mediante **author ID**.

2. El bucle exterior muestra el primer nombre y el número de cada autor, el estado y la identificación.

3. El **Recordset** agregado para cada fila se recupera de la colección **Fields** y se asignan a *rsChapter*.

4. El bucle interno muestra cuatro campos de cada fila en el **Recordset** agregado.

Ejemplo

```
Sub datashape()

    Dim cnn As New ADODB.Connection

    Dim rst As New ADODB.Recordset

    Dim rsChapter As Variant

    cnn.Provider = "MSDataShape"

    cnn.Open "Data Provider=MSDASQL;" &__
"DSN=vfox;uid=sa;pwd=vfox;database=pubs”

'PASO 1

    rst.StayInSync = FALSE

    rst.Open "SHAPE  {select * from authors}
APPEND ({select * from titleauthor} AS chapter
RELATE au_id TO au_id)",
cnn

'PASO 2

    While Not rst.EOF

        Debug.Print rst("au_fname"), rst("au_lname"),
rst("state"), rst("au_id")

'PASO 3
```

```
        rsChapter = rst("chapter")

'PASO 4

        While Not rsChapter.EOF

            Debug.Print rsChapter(0), rsChapter(1),
rsChapter(2), rsChapter(3)

            rsChapter.MoveNext

        Wend

        rst.MoveNext

    Wend

End Sub
```

Proveedores necesarios para la formación de datos

Un proveedor de origen de datos de base de datos OLE debe proporcionar datos que se pasarán a otro proveedor, que llevará a cabo la formación de datos. El proveedor de origen de datos se especifica en la cadena de conexión del objeto **Connection** como "Shape Provider=*SuOrigenDeDatos*". El proveedor que suministra el soporte de formación de datos se especifica en la propiedad **Provider** del objeto **Connection** como "MSDataShape".

Ejemplo

```
Dim cnn As New ADODB.Connection

cnn.Provider = "MSDataShape"

cnn.Open "Shape
Provider=MSDASQL;DSN=vfox;uid=sa;pwd=vfox;database=pubs”
```

Comando Compute de Shape

Un comando *COMPUTE* de Shape ejecuta una función agregada en las filas del **Recordset** secundario para generar un **Recordset** principal y, a continuación, asigna el **Recordset** secundario a la propiedad **Value** de los objetos **Field** en el **Recordset** principal generado.

Sintaxis

"SHAPE {*child-command*} [[AS] *table-alias*]

COMPUTE *aggregate-command-field-list*

[BY *grp-field-list*]"

Descripción de las partes

Las partes de este comando son:

- *child-command*: Un comando de consulta que devuelve un objeto **Recordset**. El comando se emite al proveedor de datos principal, y su sintaxis depende de los requisitos de ese proveedor. Normalmente ésta será SQL, aunque ADO no necesita el uso de ningún lenguaje de consulta en particular.
- *table-alias*: Un alias utilizado para hacer referencia al **Recordset** devuelto por *child-command.*
- *aggregate-command-field-list*: Una lista de los campos en los que opera una función agregada. Se puede hacer referencia al **Recordset** mediante su *alias de tabla* en la función agregada. También puede crear una columna para utilizarla como desee, con el operador NEW.

Función Agregar	**Descripción**
SUM(<alias>.<nombre de campo secundario>)	Calcula la suma de todos los valores en el campo especificado.
AVG(<alias>.<nombre de campo secundario>)	Calcula la media de todos los valores en el campo especificado.
MAX(<alias>.<nombre de campo secundario>)	Calcula el valor máximo en el campo especificado.
MIN(<alias>.<nombre de campo secundario>)	Calcula el valor mínimo en el campo especificado.
COUNT(<alias>[.<nombre de campo secundario>])	Cuenta el número de filas en el campo especificado.

STDEV(<alias>.<nombre de campo de nivel inferior>)	Calcula la desviación en el campo especificado.
ANY(<alias>.<nombre de campo secundario>)	El valor de una columna (donde el valor de la columna es el mismo para todas las filas).
CALC(expresión)	Calcula una expresión arbitraria, pero sólo en la fila actual.
NEW (tipo de campo [(ancho | escala [,precisión])]	Añade una columna vacía del tipo especificado al **Recordset**.

- *lista de campos de grupo*: Una lista de columnas que especifica el orden de las filas en el **Recordset** de nivel inferior. Si la cláusula **BY** no se especifica, entonces sólo se devolverá el resultado de la función agregada. Si se especifica la cláusula **BY**, el **Recordset** secundario se agregará al Recordset principal generado.

Operación

El motor de cursor cliente emitirá el *child-command* al proveedor, que devolverá un **Recordset** secundario.

La cláusula COMPUTE especifica una operación agregada para que se ejecute en las columnas especificadas (*lista de campos del comando agregado*) del **Recordset** secundario, como sumar todos los valores en una columna, o encontrar el valor máximo de una columna. La operación de agregación crea un **Recordset** principal.

Si no existe una cláusula **BY**, entonces el comando Shape concluye. Si existe una cláusula **BY**, el **Recordset** secundario se agregará al Recordset principal. Las filas del **Recordset** secundario serán dispuestas en grupos según se especifica en *lista de campos de grupo*.

Por ejemplo, suponga que tiene una tabla, Demografía, que contiene los campos Estado, Ciudad y Población (las cifras de población son sólo ilustrativas).

Estado	**Ciudad**	**Población**
WA	Seattle	700.000
OR	Medford	200.000
OR	Portland	600.000
CA	Los Angeles	900.000
CA	San Diego	400.000
WA	Tacoma	500.000
OR	Corvallis	300.000

Ahora, emita este comando Shape:

```
    rst.Open "SHAPE  {select * from demografía}
COMPUTE (SUM(population)) AS chapter
BY estado",
connection
```

Este comando abre un **Recordset** principapl. Como se ha especificado la cláusula **BY**, se ha añadido una columna oculta, con referencias a objetos **Recordset** que proporciona detalle para cada fila en el **Recordset** principal.

Se agruparán las filas de detalle del **Recordset** secundario, pero sin un orden en particular. Es decir, los grupos no se colocarán automáticamente en orden alfabético o numérico.

Se generará un nombre arbitrario para la columna agregada.

Ahora puede explorar por el **Recordset** principal abierto y tener acceso a los objetos del **Recordset** secundario de detalle. Consulte Tener acceso a las filas de un Recordset jerárquico.

Recordsets de detalle primarios y secundarios resultantes

Primarios

Estado	SUM (Población)	(Campo agregado, oculto)
CA	1.300.000	Referencia a secundario 1
WA	1.200.000	Referencia a secundario 2
OR	1.100.000	Referencia a secundario 3

Secundario 1

Estado	Ciudad	Población
CA	Los Angeles	900.000
CA	San Diego	400.000

Secundario 2

Estado	Ciudad	Población
WA	Seattle	700.000
WA	Tacoma	500.000

Secundario 3

Estado	Ciudad	Población
OR	Medford	200.000
OR	Portland	600.000
OR	Corvallis	300.000

Comando Append de Shape

El comando *APPEND* de Shape asigna un **Recordset** secundario a la propiedad **Value** de objetos **Field** en un **Recordset** principal.

Sintaxis

"SHAPE {*parent-command*} [[AS] *table-alias*]

APPEND {*child-command*}

RELATE(*parent-column* TO *child-column*)"

Descripción de las partes

Las partes de este comando son:

- *parent-command*, *child-command*: Un comando de consulta que devuelve un objeto **Recordset**. El comando se emite al proveedor de datos principal y su sintaxis depende de los requisitos de ese proveedor. Normalmente ésta será SQL, aunque ADO no necesita el uso de ningún lenguaje de consulta en particular.

- *parent-column*: Una columna del **Recordset** devuelto por un *parent-command.*

- *child-column*: Una columna del **Recordset** devuelto por un *child-command.*

- *table-alias*: Un alias utilizado para referirse al **Recordset** devuelto por *parent-command.*

Operación

El motor de cursor cliente emitirá el *parent-command* al proveedor, que devolverá un **Recordset** principal. Posteriormente se emite el *child-command*, que devuelve un **Recordset** secundario.

Por ejemplo, *parent-command* podría devolver un **Recordset** de clientes para una compañía de una base de datos Clientes y el *child-command* podría devolver un Recordset de pedidos para todos los clientes a partir de una base de datos Pedidos.

Los objetos **Recordset** secundarios y principales deben tener una columna en común. A las columnas se les asigna un nombre en la cláusula **RELATE**, primero *parent-command*, después *child-column*. Las columnas pueden tener nombres distintos en los respectivos objetos **Recordset**, pero deben referirse a la misma información para especificar una relación con sentido.

Por ejemplo, los recordset Clientes y Pedido podrán tener en común los campos CustomerID y BuyerID.

El motor de cursor cliente crea internamente una nueva columna y la agrega literalmente al **Recordset** principal. Los valores de los campos en la nueva columna son referencias a las filas del recordset secundario que satisfacen la cláusula **RELATE**.

La columna añadida se llamará automáticamente "chapter" y será del tipo de datos **adChapter**. Si desea explorar el **Recordset** secundario, especifique la columna agregada en la colección **Fields** del objeto **Recordset** y recupere el **Recordset** desde la propiedad **Value** del objeto **Field**. Asigne el **Recordset** recuperado a un objeto **Recordset** vacío y, a continuación, explore ese **Recordset** como lo haría con cualquier otro.

Comandos parametrizados

Los comandos **Shape** se pueden parametrizar. Por ejemplo, puede especificar lo siguiente:

```
"SHAPE {SELECT * FROM customer}
APPEND {SELECT * FROM orders WHERE cust_id = ?}
RELATE (cust_id TO PARAMETER 0)"
```

En este caso, la tablas principal y secundaria se producen para que tengan un nombre de columna en común, *cust_id.* El *child-command* tiene un marcador de posición (esto es, "**?**"), al que se refiere la cláusula **RELATE** (esto es, "...PARAMETER 0". En efecto, la relación se da entre la *parent-column* identificada explícitamente y la *child-column* identificada implícitamente por el marcador de posición).

Cuando el comando Shape se ejecuta, lo que ocurre en realidad es:

- Se agrega un columna vacía al **Recordset** principal.
- El *parent-command* se ejecuta y devuelve una fila de la tabla **customer**.
- El valor de la columna **customer.cust_id** reemplaza al marcador de posición y se ejecuta el *child-command*.
- Todas la filas de la tabla **orders** donde la columna **orders.cust_id** coincida con la columna **customer.cust_id** se recuperan.
- Se coloca una referencia a las filas secundarias recuperadas en la fila actual de la columna agregada al **Recordset** principal.
- A continuación, la siguiente fila principal se recupera de la tabla **customer** y se repite el ciclo hasta que se recuperan todas las filas.