SELECT (Instrucción)

Pide al motor de base de datos Microsoft Jet que devuelva información de la base de datos como un conjunto de registros.
Sintaxis:

```
SELECT [predicado] { * | tabla.* | [tabla.]campo1 [AS alias1] [, [tabla.]
campo2 [AS alias2] [, ...]]}
FROM expresiónTabla [, ...] [IN BasedeDatosExterna]
[WHERE... ]
[GROUP BY... ]
[HAVING... ]
[ORDER BY...
[WITH OWNERACCESS OPTION]
```

La **sintaxis** de la instrucción SELECT consta de las siguientes partes:

Parte	Descripción
Predicado	Uno de los siguientes predicados: ALL, DISTINCT, DISTINCTROW o TOP. Puede utilizar el predicado para limitar el número de registros devueltos. Si no especifica ninguno, el valor predeterminado es ALL.
*****	Especifica que se seleccionan todos los campos de la tabla o tablas especificadas.
Tabla	El nombre de la tabla que contiene los campos de la que se van a seleccionar los registros.
campo1, campo2	Los nombres de los campos que contienen los datos que desea recuperar. Si incluye más de un campo, éstos se recuperan en el orden enumerado.
alias1, alias2	Los nombres que se van a utilizar como encabezados de columnas en vez de los nombres de columnas originales en tabla.
ExpresiónTabla	El nombre de la tabla o las tablas que contienen los datos que desea recuperar.
BasededatosExterna	El nombre de la base de datos que contiene las tablas en ExpresiónTabla si no están en la base de datos activa.

Comentarios:
Para realizar esta operación, el motor de base de datos Microsoft Jet busca la tabla o las tablas especificadas, extrae las columnas elegidas, selecciona las filas que cumplen los criterios y ordena o agrupa las filas resultantes en el orden especificado.
Las instrucciones SELECT no modifican los datos de la base de datos.
La instrucción SELECT suele ser la primera palabra en una instrucción SQL. La mayoría de las instrucciones SQL son instrucciones SELECT o SELECT...INTO.

La **sintaxis mínima** para una instrucción **SELECT** es:

SELECT Campos FROM Tabla

Puede utilizar un asterisco (*****) para seleccionar todos los campos de una tabla. El ejemplo siguiente selecciona todos los campos de la tabla Empleados:

SELECT * FROM Empleados;

Si se incluye un nombre de campo en más de una tabla en la instrucción FROM, escriba delante el nombre de la tabla y el operador . (punto). En el siguiente ejemplo, el campo Departamento está en la tabla Empleados y en la tabla Supervisores.

Ejemplo: La instrucción SQL selecciona los departamentos de la tabla Empleados y los nombres de supervisores de la tabla Supervisores:

```
SELECT Empleados.Departamento, Supervisores.NombreSupervisor
FROM Empleados INNER JOIN Supervisores
WHERE Empleados.Departamento = Supervisores.Departamento;
```

Cuando se crea un objeto Recordset, el motor de base de datos Microsoft Jet utiliza el nombre de campo de la tabla como el nombre de objeto Field en el objeto Recordset. Si desea un nombre de campo diferente o un nombre que no participe en la expresión utilizada para generar el campo, utilice la palabra reservada AS.

El **siguiente ejemplo** utiliza el título Nacimiento para dar nombre al objeto Field devuelto en el objeto Recordset resultante:

```
SELECT FechaNacimiento
AS Nacimiento FROM Empleados;
```

Siempre que utilice funciones de dominio agregado o consultas que devuelvan nombres de objetos Field ambiguos o duplicados, debe utilizar la cláusula AS para proporcionar un nombre alternativo para el objeto Field.
El **siguiente ejemplo** utiliza el título CuentaEncabezado para dar nombre al objeto Field devuelto del objeto Recordset resultante:

```
SELECT COUNT(IdEmpleado)
AS CuentaEncabezado FROM Empleados;
```

Puede utilizar otras cláusulas en una instrucción SELECT para limitar y organizar aún más los datos devueltos. Para obtener más información al respecto, vea el tema de Ayuda relativo a las cláusulas que esté utilizando.

ALL, DISTINCT, DISTINCTROW, TOP (Predicados)

Especifican los registros seleccionados en las consultas SQL.
Sintaxis:

```
SELECT [ALL | DISTINCT | DISTINCTROW | [TOP n [PERCENT]]]
FROM tabla
```

Una instrucción SELECT que contiene estos predicados consta de las siguientes partes:

Parte	Descripción

ALL
El valor asumido si no incluye ninguno de los predicados. El motor de base de datos Microsoft Jet selecciona todos los registros que cumplen las condiciones de la instrucción SQL.
Los dos ejemplos siguientes son equivalentes y devuelven todos los registros de la tabla Empleados:

```
SELECT ALL * FROM Empleados
ORDER BY IdEmpleado;

SELECT * FROM Empleados
ORDER BY IdEmpleado;
```

DISTINCT
Omite los registros que contienen datos duplicados en los campos seleccionados. Para que se incluyan en el resultado de la consulta, los valores para cada campo enumerado en la instrucción SELECT debe ser único. **Por ejemplo:** Varios empleados contenidos en una tabla Empleados pueden tener el mismo apellido. Si dos registros contienen Alonso en el campo Apellidos, la instrucción SQL siguiente devuelve solamente un registro que contiene Alonso:

```
SELECT DISTINCT Apellidos
FROM Empleados;
```

Si omite el predicado DISTINCT, esta consulta devuelve los dos registros Alonso.
Si la cláusula SELECT contiene más de un campo, la combinación de valores de todos los campos debe ser única para un registro concreto que se va a incluir en el resultado.El resultado de una consulta que utiliza el predicado DISTINCT no se puede actualizar y no refleja cambios subsiguientes realizados por otros usuarios.

DISTINCTROW
Omite los datos basados en registros duplicados completos, no sólo campos duplicados. **Por ejemplo**, puede crear una consulta que combine las tablas Clientes y Pedidos por el campo IdCliente. La tabla Clientes no contiene ningún campo IdCliente duplicado, pero la tabla Pedidos sí los tiene porque cada cliente puede tener varios pedidos. La instrucción SQL siguiente muestra cómo puede utilizar el predicado DISTINCTROW para crear una lista de compañías que tengan al menos un pedido pero sin obtener detalles acerca de los mismos:

```
SELECT DISTINCTROW NombreCompañía FROM Clientes INNER JOIN Pedidos
ON Clientes. IdCliente = Pedidos.IdCliente ORDER BY NombreCompañía;
```

Si omite el predicado DISTINCTROW, esta consulta crea múltiples filas por cada compañía que tenga más de un pedido.El predicado DISTINCTROW tiene efecto solamente cuando selecciona campos desde algunas tablas, pero no todas, utilizadas en la consulta. El predicado DISTINCTROW se ignora si su consulta incluye solamente una tabla o si da salida a campos de todas las tablas.

TOP n [PERCENT]
Devuelve cierto número de registros que se encuentran entre la parte superior e inferior de un intervalo especificado por una cláusula ORDER BY.
Ejemplo: Suponga que desea obtener los nombres de los primeros 25 alumnos de la promoción de 1994:

```
SELECT TOP 25 Nombre, Apellidos
FROM Alumnos  WHERE AñoGraduación = 1994
ORDER BY NotasMedias DESC;
```

Si no incluye la cláusula ORDER BY, el resultado de la consulta devolverá un conjunto aleatorio de 25 registros de la tabla Alumnos que cumplan la cláusula WHERE. El predicado TOP no elige entre valores iguales. En el ejemplo anterior, si las notas medias de los registros número veinticinco y veintiséis son iguales, la consulta devolverá 26 registros. También puede utilizar la palabra reservada PERCENT para devolver un determinado porcentaje de registros que se encuentran entre la parte superior e inferior de un intervalo especificado en una cláusula ORDER BY.
Ejemplo: Suponga que, en lugar de los 25 primeros estudiantes, desea el último 10 por ciento de la clase:

```
SELECT TOP 10 PERCENT
Nombre, Apellidos
FROM Alumnos
WHERE AñoGraduación = 1994
ORDER BY NotasMedias ASC;
```

El predicado ASC especifica una devolución de los valores inferiores. El valor que sigue a TOP debe se un número tipo Integer sin signo.Al predicado TOP no le afecta si la consulta es o no se puede actualizar.

Tabla El nombre de la tabla de la que se van a recuperar los registros.

FROM (Cláusula)

Especifica las tablas o consultas que contienen la lista de campos enumerados en la instrucción SELECT.
Sintaxis:

```
SELECT  Listacampos
FROM ExpresiónTabla [IN BasedeDatosExterna]
```

Una instrucción SELECT que contiene una cláusula FROM consta de las siguientes partes:

Parte	Descripción

Listacampos
El nombre del campo o de los campos que se van a recuperar junto con cualquier alias de nombre de campo, funciones de dominio agregado de SQL, predicados de selección (ALL, DISTINCT, DISTINCTROW o TOP) u otras opciones de la instrucción SELECT.

Expresióntabla
Una expresión que identifica a una o más tablas de las que se van a recuperar datos. La expresión puede ser un nombre de tabla único, un nombre de consulta guardada o una composición resultante de una cláusula INNER JOIN, LEFT JOIN o RIGHT JOIN.

BasedeDatosExterna
La ruta completa de una base de datos externa que contiene todas las tablas en ExpresiónTabla.

Comentarios:

La cláusula FROM es obligatoria y sigue a cualquier instrucción SELECT. El orden de los nombres de las tablas en ExpresiónTabla no es importante. Para mejorar el rendimiento y facilitar el uso, se recomienda que utilice una tabla vinculada en lugar de una cláusula IN para recuperar datos de una base de datos externa.

El **siguiente ejemplo** muestra cómo puede recuperar datos de la tabla Empleados:

```
SELECT Apellidos, Nombre
FROM Empleados;
```

Funciones de dominio agregado de SQL

Utilizando las funciones de dominio agregado de SQL puede determinar varias estadísticas sobre conjuntos de valores. Puede utilizar estas funciones en cualquier consulta y expresión de agregado en la propiedad SQL de un objeto QueryDef o cuando crea un objeto Recordset basado en una consulta SQL.

Función Avg
Función Count
Funciones Min, Max
Funciones StDev, StDevP
Función Sum
Funciones Var, VarP

Avg (Función)

Calcula la media aritmética de un conjunto de valores contenido en un campo especificado de una consulta.

Sintaxis:

Avg(expresión)

El marcador de posición expresión representa una expresión de cadena que identifica el campo que contiene los datos numéricos de los que desea obtener la media o una expresión que ejecuta un cálculo utilizando los datos de ese campo. Los operandos de expresión pueden incluir el nombre del campo de una tabla, una constante o una función (que puede ser intrínseca o definida por el usuario, pero ninguna de las demás funciones de dominio agregado de SQL).

Comentarios:

La media calculada por la función Avg es la media aritmética (la suma de los valores dividido entre el número de valores). Podría utilizar la función Avg, por ejemplo, para calcular el costo medio de envío.
La función Avg no incluye ningún campo Null en el cálculo.
Puede utilizar la función Avg en una expresión de consulta y en la propiedad SQL de un objeto QueryDef o cuando crea un objeto Recordset basado en una consulta SQL.

Avg (Ejemplo de la función)

Este ejemplo utiliza la tabla Pedidos para calcular el promedio de los cargos por envío por pedidos cuyos cargos por envío superen los 100 dólares.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub AvgX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' calcular el promedio de los cargos por envío por
    ' pedidos cuyos cargos por envío superen los 100 dólares.
    Set rst = dbs.OpenRecordset("SELECT Avg(Cargo)" _
        & "AS [Promedio cargo]" _
        & "FROM Pedidos WHERE Cargo > 100;")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
    ' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 25
    dbs.Close
End Sub
```

Count (Función)

Calcula el número de registros devueltos por una consulta.

Sintaxis:

Count(expresión)

El marcador de posición expresión representa una expresión de cadena que identifica el campo que contiene los datos que desea contar o una expresión que ejecuta un cálculo utilizando los datos de ese campo. Los operandos de expresión pueden incluir el nombre del campo de una tabla o una función (que puede ser intrínseca o definida por el usuario, pero ninguna de las demás funciones de dominio agregado de SQL). Puede contar cualquier tipo de datos, incluido tipo texto.

Comentarios:

Puede utilizar la función Count para contar el número de registros de una consulta base. Por ejemplo, podría utilizar la función Count para contar el número de pedidos enviados a un país determinado.

Aunque expresión puede ejecutar un cálculo sobre un campo, la función Count simplemente mide el número de registros. No importan los valores que están almacenados en los registros. La función Count no cuenta registros que tengan campos Null a menos que expresión sea el carácter comodín asterisco (*). Si utiliza un asterisco, la función Count calcula el número total de registros, incluyendo los que contienen campos Null. Count(*) es considerablemente más rápido que Count([Nombre de columna]). No escriba el asterisco entre comillas (' ').

El **siguiente ejemplo** calcula el número de registros de la tabla Pedidos:

```
SELECT Count(*)
AS TotalPedidos FROM Pedidos;
```

Si expresión identifica múltiples campos, la función Count cuenta un registro solamente si al menos uno de los campos no es Null. Si todos los campos especificados son Null, no se cuenta el registro. Separe los nombres de los campos con un símbolo &.

El **siguiente ejemplo** muestra cómo puede limitar la cuenta de registros a aquellos en los que los valores de FechaEnvío o FechaEntrega no sean Null:

```
SELECT
Count('FechaEnvío & FechaEntrega')
AS [Not Null] FROM Pedidos;
```

Puede utilizar la función Count en una expresión de consulta. También puede usar esta expresión en la propiedad de SQL de un objeto QueryDef o cuando crea un objeto Recordset basado en una consulta SQL.

Count (Ejemplo de la función)

Este ejemplo utiliza la tabla Pedidos para calcular el número de pedidos enviados a España.

Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub CountX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Calcula el número de pedidos enviados
    ' a España.
    Set rst = dbs.OpenRecordset("SELECT" _
        & "Count (País destinatario)" _
        & "AS [Pedidos a España] FROM Pedidos" _
        & "WHERE País destinatario = 'España';")
    ' Llena el Recordset.
    rst.MoveLast
' Llama a EnumCampos para imprimir el contenido del
    ' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 25
    dbs.Close
End Sub
```

Min, Max (Funciones)

Devuelven el valor mínimo o máximo de un conjunto de valores contenidos en un campo especificado de una consulta.

Sintaxis:

Min(expresión)
Max(expresión)

El marcador de posición expresión representa una expresión de cadena que identifica el campo que contiene los datos que desea calcular o una expresión que ejecuta un cálculo utilizando los datos de ese campo. Los operandos de expresión pueden incluir el nombre del campo de una tabla, una constante o una función (que puede ser intrínseca o definida por el usuario, pero ninguna de las demás funciones de dominio agregado de SQL).

Comentarios:

Puede utilizar las funciones Min y Max para determinar los valores mayor y menor de un campo basándose en el agregado o agrupamiento especificado. Por ejemplo, podría utilizar estas funciones para devolver el menor y el mayor costo de envío. Si no se especifica ningún agregado, se utiliza toda la tabla. Puede utilizar las funciones Min y Max en una expresión de consulta y en la propiedad de SQL de un objeto QueryDef o cuando crea un objeto Recordset basado en una consulta SQL.

Min, Max (Ejemplo de la funciones)

Este ejemplo utiliza la tabla Pedidos para devolver el cargo por envío mayor y menor de los pedidos enviados a España.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub MinMaxX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Devuelve el cargo por envío mayor y menor de los
    ' pedidos enviados a España.
    Set rst = dbs.OpenRecordset("SELECT " _
        & "Min(Cargo) AS [Cargo menor], " _
        & "Max(Cargo)AS [Cargo mayor] " _
        & "FROM Pedidos WHERE País destinatario = 'España';")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
    ' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 12
    dbs.Close
End Sub
```

StDev, StDevP (Funciones)

Devuelve n estimaciones de la desviación estándar de una población o una muestra de población representada como un conjunto de valores contenidos en un campo especificado de una consulta.

Sintaxis:

StDev(expresión)
StDevP(expresión)

El marcador de posición expresión representa una expresión de cadena que identifica el campo que contiene los datos numéricos que desea calcular o una expresión que ejecuta un cálculo utilizando los datos de ese campo. Los operandos de expresión pueden incluir el nombre del campo de una tabla, una constante o una función (que puede ser intrínseca o definida por el usuario, pero ninguna de las demás funciones de dominio agregado de SQL).

Comentarios:

La función StDevP evalúa una población y la función StDev evalúa una muestra de población.
Si la consulta base contiene menos de dos registros (o no contiene registros, para la función StDevP), estas funciones devuelven un valor Null (que indica que no se puede calcular la desviación estándar). Puede utilizar las funciones StDev y StDevP en una expresión de consulta. También puede utilizar esta expresión en la propiedad de SQLde un objeto QueryDef o cuando crea un objeto Recordset basado en una consulta SQL.

StDev, StDevP (Ejemplo de la funciones)

Este ejemplo utiliza la tabla Pedidos para estimar la desviación estándar de los cargos por envío de pedidos enviados a España.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub StDevX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Calcula la desviación estándar de los cargos por
    ' pedidos de los pedidos enviados a España.
    Set rst = dbs.OpenRecordset("SELECT " _
        & "StDev(Cargo) " _
        & "AS [Desviación de Cargo] FROM Pedidos " _
        & "WHERE País destinatario = 'España';")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
    ' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 15
    Debug.Print
    Set rst = dbs.OpenRecordset("SELECT " _
        & "StDevP(Cargo) " _
        & "AS [Desviación P de Cargo] FROM Pedidos " _
        & "WHERE País destinatario = 'España';")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 15
    dbs.Close
End Sub
```

Sum (Función)

Devuelve la suma de un conjunto de valores contenidos en un campo especificado de una consulta.

Sintaxis:

Sum(expresión)

El marcador de posición expresión representa una expresión de cadena que identifica el campo que contiene los datos numéricos que desea sumar o una expresión que ejecuta un cálculo utilizando los datos de ese campo. Los operandos de expresión pueden incluir el nombre del campo de una tabla, una constante o una función (que puede ser intrínseca o definida por el usuario, pero ninguna de las demás funciones de dominio agregado de SQL).

Comentarios:

La función Sum totaliza los valores de un campo. Por ejemplo, podría utilizar la función Sum para determinar el costo total de los gastos de envío. La función Sum ignora los registros que contienen campos Null. El **siguiente ejemplo** muestra cómo calcular la suma de los productos de los campos PrecioUnidad y Cantidad:

```
SELECT Sum(PrecioUnidad * Cantidad)
AS [Total Ingresos] FROM [Detalles de pedidos];
```

Puede utilizar la función Sum en una expresión de consulta. También puede usar esta expresión en la propiedad de SQL de un objeto QueryDef o cuando crea un objeto Recordset basado en una consulta SQL.

Sum (Ejemplo de la función)

Este ejemplo utiliza la tabla Pedidos para calcular las ventas totales de pedidos enviados a España.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub SumX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
```

```
        Set dbs = OpenDatabase("Neptuno.mdb")
        ' Calcula las ventas totales de pedidos enviados a
        ' España.
        Set rst = dbs.OpenRecordset("SELECT" _
                & "Sum(PrecioUnidad*Cantidad)" _
                & "AS [Total España Ventas] FROM Pedidos" _
                & "INNER JOIN [Detalles de pedidos] ON" _
                & "Pedidos.IdPedido = [Detalles de pedidos].IdPedido" _
                & "WHERE (País destinatario = 'España');")
        ' Llena el Recordset.
        rst.MoveLast
        ' Llama a EnumCampos para imprimir el contenido del
        ' Recordset. Transfiere el objeto Recordset y el
        ' tamaño de campo deseado.
        EnumCampos rst, 15
        dbs.Close
End Sub
```

Var, VarP (Funciones)

Devuelve n estimaciones de la varianza de una población o una muestra de población representada como un conjunto de valores contenidos en un campo especificado de una consulta.

Sintaxis:

Var(expresión)
VarP(expresión)

El marcador de posición expresión representa una expresión de cadena que identifica el campo que contiene los datos numéricos que desea evaluar o una expresión que ejecuta un cálculo utilizando los datos de ese campo. Los operandos de expresión pueden incluir el nombre del campo de una tabla, una constante o una función (que puede ser intrínseca o definida por el usuario, pero ninguna de las demás funciones de dominio agregado de SQL).

Comentarios:

La función VarP evalúa una población y la función Var evalúa una muestra de población.
Si la consulta base contiene menos de dos registros, las funciones Var y VarP devuelven un valor Null, que indica que no se puede calcular la varianza. Puede utilizar las funciones Var y VarP en una expresión de consulta o en una instrucción SQL.

Var, VarP (Ejemplo de la funciones)

Este ejemplo utiliza la tabla Pedidos para estimar la varianza de los cargos por envío de pedidos enviados a España.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub VarX()
        Dim dbs As Database, rst As Recordset
        ' Modifique esta línea para incluir la ruta de
        ' acceso a la base de datos Neptuno en su equipo.
        Set dbs = OpenDatabase("Neptuno.mdb")
        ' Calcula la varianza de los cargos por envío de
        ' pedidos enviados a España.
        Set rst = dbs.OpenRecordset("SELECT " _
                & "Var(Cargo) " _
                & "AS [Varianza de cargos a España] " _
                & "FROM Pedidos WHERE País destinatario = 'España';")
        ' Llena el Recordset.
          rst.MoveLast
        ' Llama a EnumCampos para imprimir el contenido del
        ' Recordset. Transfiere el objeto Recordset y el
        ' tamaño de campo deseado.
        EnumCampos rst, 20
        Debug.Print
        Set rst = dbs.OpenRecordset("SELECT " _
                & "VarP(Cargo) " _
                & "AS [VarianzaP de cargos a España] " _
                & "FROM Pedidos WHERE País destinatario = 'España';")
        ' Llena el Recordset.
        rst.MoveLast
        ' Llama a EnumCampos para imprimir el contenido del
```

```
        ' Recordset. Transfiere el objeto Recordset y el
        ' tamaño de campo deseado.
        EnumCampos rst, 20
        dbs.Close
End Sub
```

Calcular campos en funciones SQL

Puede utilizar el argumento expresión de cadena en una función de dominio agregado de SQL para realizar un cálculo sobre valores de un campo. Por ejemplo, podría calcular un porcentaje (como un recargo o un impuesto sobre ventas) mediante la multiplicación del valor de un campo por una fracción.
La siguiente tabla proporciona **ejemplos** de cálculos sobre campos de las tablas Pedidos y Detalles de pedidos en la base de datos Neptuno.mdb.

Cálculo	**Ejemplo**
Sumar un número a un campo	Cargo + 5
Restar un número de un campo	Cargo - 5
Multiplicar un campo por un número	PrecioUnidad * 2
Dividir un campo por un número	Cargo / 2
Sumar un campo a otro	UnidadesEnExistencia + UnidadesEnPedido
Restar un campo de otro	NivelNuevoPedido – UnidadesEnExistencia

El **siguiente ejemplo** calcula el importe del descuento medio de todos los pedidos de la base de datos Neptuno.mdb. Multiplica los valores de los campos PrecioUnidad y Descuento para determinar el importe del descuento de cada pedido y después calcula la media. Puede utilizar esta expresión en una instrucción SQL en código de Visual Basic:

```
SELECT Avg(PrecioUnidad * Descuento) AS [Descuento medio] FROM [Detalles de pedidos];
```

GROUP BY (Cláusula)

Combina en un único registro los registros con valores idénticos en la lista de campos especificada. Se creará un valor de resumen para cada registro si incluye una función agregada de SQL, como Sum o Count, en la instrucción SELECT.
Sintaxis:

```
SELECT Listacampos FROM tabla
WHERE criterio
[GROUP BY grupolistacampos]
```

Una instrucción SELECT que contiene una cláusula GROUP BY consta de las siguientes partes:

Parte	**Descripción**

Listacampos
El nombre del campo o de los campos que se van a recuperar junto con cualquier alias de nombre de campo, predicado de selección (ALL, DISTINCT, DISTINCTROW o TOP) u otras opciones de la instrucción SELECT.

Tabla
El nombre de la tabla o tablas de las que se van a recuperar los datos. Para obtener más información al respecto, vea la cláusula FROM.

Criterio
Criterio de selección. Si la instrucción incluye una cláusula WHERE, el motor de base de datos Microsoft Jet agrupa los valores después de aplicar las condiciones de la cláusula WHERE a los registros.

Grupolistacampos
Los nombres de hasta 10 campos que se van a utilizar para agrupar registros. El orden de los nombres de los campos en grupolistacampos determina los niveles de agrupamiento desde el más alto hasta el más bajo.

Comentarios:

La cláusula GROUP BY es opcional.
Los valores de resumen se omiten si no hay ninguna función agregada de SQL en la instrucción SELECT.
Los valores Null en los campos de GROUP BY se agrupan y no se omiten. Sin embargo, los valores Null no se calculan en ninguna función agregada de SQL.
Utilice la cláusula WHERE para excluir las filas por las que no desea agrupar, y utilice la cláusula HAVING para filtrar los registros una vez agrupados.
A menos que contenga datos tipo Memo u Objeto OLE, un campo de la lista de campos GROUP BY puede hacer referencia a cualquier campo de cualquier tabla enumerada en la cláusula FROM, incluso si el campo no se incluye en la instrucción SELECT,

suponiendo que la instrucción SELECT incluya al menos una función agregada de SQL. El motor de base de datos Microsoft Jet no puede agrupar por campos tipo Memo u Objeto OLE.
Todos los campos incluidos en la lista de campos de SELECT se deben incluir en la cláusula GROUP BY o se deben incluir como argumentos en una función agregada de SQL.

GROUP BY (Ejemplo de la cláusula)

Este ejemplo crea una lista de cargos única y el número de empleados que hay en cada cargo.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub GroupByX1()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Para cada cargo, cuenta el número de empleados
    ' con ese cargo.
    Set rst = dbs.OpenRecordset("SELECT Cargo, " _
        & "Count([Cargo]) AS Cuenta " _
        & "FROM Empleados GROUP BY Cargo;")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 25
    dbs.Close
End Sub
```

Para cada cargo único, **este ejemplo** calcula el número de empleados en Washington que tienen ese cargo.

```
Sub GroupByX2()

    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Para cada cargo, cuenta el número de empleados
    ' con ese cargo. Sólo incluye los empleados de la
    ' región Washington (WA).
    Set rst = dbs.OpenRecordset("SELECT Cargo, " _
        & "Count(Cargo) AS Cuenta " _
        & "FROM Empleados WHERE Región = 'WA' " _
        & "GROUP BY Cargo;")
    ‘ Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
    ' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 25
    dbs.Close
End Sub
```

HAVING (Cláusula)

Especifica qué registros agrupados se van a mostrar en una instrucción SELECT con una cláusula GROUP BY. Después de que la cláusula GROUP BY combina los registros, la cláusula HAVING muestra los registros agrupados por la cláusula GROUP BY que cumplen las condiciones de la cláusula HAVING.

Sintaxis:

```
SELECT listacampos
FROM tabla
WHERE criterioselección
GROUP BY grupolistacampos
[HAVING grupocriterio]
```

Una instrucción SELECT que contiene una cláusula HAVING consta de las siguientes partes:

Parte **Descripción**

Listacampos
El nombre del campo o de los campos que se van a recuperar junto con cualquier alias de nombre de campo, funciones de dominio agregado de SQL , predicado de selección (ALL, DISTINCT, DISTINCTROW o TOP) u otras opciones de la instrucción SELECT.

Tabla
El nombre de la tabla o tablas de las que se van a recuperar los datos. Para obtener más información al respecto, vea la cláusula FROM.

Criterioselección
Criterio de selección. Si la instrucción incluye una cláusula WHERE, el motor de base de datos Microsoft Jet agrupa los valores después de aplicar las condiciones de la cláusula WHERE a los registros.

Grupolistacampos
Los nombres de hasta 10 campos que se van a utilizar para agrupar registros. El orden de los nombres de los campos en grupolistacampos determina los niveles de agrupamiento desde el más alto hasta el más bajo.

Grupocriterio
Una expresión que determina qué registros agrupados se van a mostrar.

Comentarios:

La cláusula HAVING es opcional.
La cláusula HAVING es similar a WHERE, que determina qué registros se van a seleccionar. Una vez agrupados los registros con la cláusula GROUP BY, la cláusula HAVING determina qué registros se van a mostrar:

```
SELECT IdCategoría, Sum(UnidadesEnExistencia)
FROM Productos
GROUP BY IdCategoría
HAVING Sum(UnidadesEnExistencia) > 100 And Like "BOS*";
```

Una cláusula HAVING puede contener hasta 40 expresiones vinculadas por operados lógicos, como And y Or.

HAVING (Ejemplo de la cláusula)

Este ejemplo selecciona el cargo asignado a más de un empleado en la región de Washington.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub HavingX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Selecciona el cargo asignado a más de un
    ' empleado en la región de Washington (WA).
    Set rst = dbs.OpenRecordset("SELECT Cargo, " _
        & "Count(Cargo) as Total FROM Empleados " _
        & "WHERE Región = 'WA' " _
        & "GROUP BY Cargo HAVING Count(Cargo) > 1;")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del recordset.
    EnumCampos rst, 25
    dbs.Close
End Sub
```

IN (Cláusula)

Identifica las tablas en cualquier base de datos externa a la que el motor de base de datos Microsoft Jet se puede conectar, como una base de datos dBASE o Paradox, o una base de datos externa de Microsoft Jet.
Sintaxis:

- ✓ **Para identificar una tabla de destino:**

```
[SELECT | INSERT] INTO destino IN
    {ruta| ["ruta " "tipo"] | ["" [tipo; DATABASE = ruta]]}
```

- ✓ **Para identificar una tabla de origen:**

```
FROM expresióntabla IN
        {ruta | ["ruta" "tipo"] | ["" [tipo; DATABASE = ruta]]}
```

Una instrucción SELECT que contiene una cláusula IN consta de las siguientes partes:

Parte	Descripción
Destino	El nombre de la tabla externa en la que se van a insertar los datos.
Expresióntabla	El nombre de la tabla o de las tablas de las que se van a recuperar los datos. Este argumento puede ser un nombre de tabla único, una consulta guardada o una composición resultante de una cláusula INNER JOIN, LEFT JOIN o RIGHT JOIN.
Ruta	La ruta completa del directorio o archivo que contiene tabla.
Tipo	El nombre del tipo de base de datos utilizado para crear tabla si no es una base de datos de Microsoft Jet (por ejemplo, dBASE III, dBASE IV, Paradox 3.x o Paradox 4.x).

Comentarios:

Puede utilizar la cláusula IN para conectarse solamente a una base de datos externa cada vez.
En algunos casos, el argumento ruta hace referencia al directorio que contiene los archivos de bases de datos. Por ejemplo, cuando trabaja con tablas de bases de datos de dBASE, FoxPro o Paradox, el argumento ruta especifica el directorio que contiene los archivos .dbf o .db. El nombre del archivo de tabla se deriva del argumento destino o expresióntabla.
Para especificar una base de datos que no sea de Microsoft Jet, agregue un punto y coma (;) al nombre y escríbalo entre comillas simples (' ') o dobles (" "). **Por ejemplo**, se aceptan 'dBASE IV;' o "dBASE IV;". También puede utilizar la palabra reservada DATABASE para especificar la base de datos externa. Por ejemplo, las siguientes líneas especifican la misma tabla:

... FROM Tabla IN "" [dBASE IV; DATABASE=C:\DBASE\DATOS\VENTAS;];
... FROM Tabla IN "C:\DBASE\DATOS\VENTAS" "dBASE IV;"

Notas:

- ✓ Para mejorar el rendimiento y facilitar el uso, utilice una tabla vinculada en lugar de la cláusula IN.
- ✓ También puede utilizar la palabra reservada IN como un operador de comparación en una expresión. Para obtener más información al respecto, vea el operador In.

IN (Ejemplo de la cláusula)

La **siguiente tabla** muestra cómo puede utilizar la cláusula IN para recuperar datos de una base de datos externa. En cada ejemplo, supone que la tabla Clientes hipotética se almacena en una base de datos externa.

Base de datos externa — **Instrucción SQL**

Base de datos de Microsoft Jet

```
SELECT IdCliente
FROM Clientes
IN OtraBD.mdb
WHERE IdCliente Like "A*";
```

dBASE III o IV. Para recuperar datos de una tabla de dBASE III, sustituya "dBASE III;" por "dBASE IV;". dBASE III o IV utilizando la sintaxis Database.

```
SELECT IdCliente
FROM Cliente
IN "C:\DBASE\DATOS\VENTAS" "dBASE IV;"
WHERE IdCliente Like "A*";

SELECT IdCliente
FROM Cliente
IN "" [dBASE IV; Database=C:\DBASE\DATOS\VENTAS;]
WHERE IdCliente Like "A*";
```

Paradox 3.x o 4.x.Para recuperar datos de una tabla de Paradox versión 3.x, sustituya "Paradox 3.x;" por "Paradox 4.x;". Paradox 3.x o 4.x utilizando la sintaxis Database.

```
SELECT IdCliente
FROM Cliente
IN "C:\PARADOX\DATOS\VENTAS" "Paradox 4.x;"
WHERE IdCliente Like "A*";

SELECT IdCliente
FROM Cliente
IN "" [Paradox 4.x;Database=C:\PARADOX\DATOS\VENTAS;]
WHERE IdCliente Like "A*";
```

Una hoja de cálculo de Microsoft Excel

```
SELECT IdCliente, Compañía
FROM [Clientes$]
IN "c:\docs\xldatos.xls" "EXCEL 5.0;"
WHERE IdCliente Like "A*"
ORDER BY IdCliente;
```

Un intervalo con nombre en una hoja de cálculo

```
SELECT IdCliente, Compañía
FROM IntervClientes
IN "c:\docs\xldata.xls" "EXCEL 5.0;"
WHERE IdCliente Like "A*"
ORDER BY IdCliente;
```

INSERT INTO (Instrucción)

Agrega uno o varios registros a una tabla. Esto se conoce como una consulta de datos añadidos.
Sintaxis:

- ✓ **Consulta de datos añadidos para múltiples registros:**

```
INSERT INTO destino [IN basededatosexterna] [(campo1[, campo2[, ...]])]
SELECT [origen.]campo1[, campo2[, ...]
FROM expresióntabla
```

- ✓ **Consulta de datos añadidos para un registro:**

```
INSERT INTO destino [(campo1[, campo2[, ...]])]
VALUES (valor1[, valor2[, ...])
```

La instrucción INSERT INTO consta de las siguientes partes:

Parte	Descripción
Destino	El nombre de la tabla o consulta donde se van a añadir registros.
Basededatosexterna	La ruta de una base de datos externa. Si desea una descripción de la ruta de acceso, vea la cláusula IN.
Origen	El nombre de la tabla o consulta de donde se van a copiar registros.
campo1, campo2	Los nombres de los campos donde se van a añadir los datos, si está a continuación del argumento destino, o los nombres de los campos de donde se obtienen los datos, si está a continuación de un argumento origen.
Expresióntabla	El nombre de la tabla o las tablas de donde se van a insertar los registros. Este argumento puede ser un nombre de tabla sencillo o uno compuesto resultante de una operación INNER JOIN, LEFT JOIN o RIGHT JOIN, o una consulta guardada.
valor1, valor2	Los valores que se van a insertar en los campos específicos del nuevo registro. Cada valor se inserta en el campo que corresponde a la posición del valor en la lista: valor1 se inserta en campo1 del nuevo registro, valor2 dentro de campo2, y así sucesivamente. Debe separar los valores con una coma y escribir los campos de texto entre comillas (' ').

Comentarios:

Puede utilizar la instrucción INSERT INTO para agregar un único registro a una tabla utilizando la sintaxis de consulta de datos añadidos para un único registro como se muestra anteriormente. En este caso, su código especifica el nombre y valor de cada campo del registro. Debe especificar cada uno de los campos del registro a los que se va a asignar un valor y dicho valor para ese campo. Cuando no especifique cada campo, se inserta el valor predeterminado o Null en las columnas no especificadas. Los registros se agregan al final de la tabla.

También puede utilizar la instrucción INSERT INTO para añadir un conjunto de registros de otra tabla o consulta utilizando la cláusula SELECT ... FROM tal y como se muestra anteriormente en la sintaxis de consulta de datos añadidos para múltiples registros. En este caso, la cláusula SELECT especifica los campos a añadir a la tabla destino especificada.
La tabla origen o destino puede especificar una tabla o una consulta. Si especifica una consulta, el motor de base de datos Microsoft Jet añade registros a cualquiera y a todas las tablas especificadas en la consulta.
La instrucción INSERT INTO es opcional pero cuando se incluye, debe preceder a la instrucción SELECT.
Si su tabla de destino contiene una clave principal, asegúrese de que añade un valor único y distinto de Null al campo o campos de la clave principal; si no, el motor de base de datos Microsoft Jet no añadirá los registros.
Si añade registros a una tabla con un campo de tipo AutoNumber y desea volver a numerar los registros añadidos, no incluya el campo AutoNumérico en su consulta. Incluya el campo Autonumérico en la consulta si desea conservar los valores originales del campo.

Utilice la cláusula IN para añadir registros a una tabla de otra base de datos.
Para crear una nueva tabla, utilice la instrucción SELECT... INTO en lugar de crear una consulta de creación de tabla.
Para ver qué registros se van a añadir antes de ejecutar la consulta de datos añadidos, ejecute primero una consulta de selección que utilice el mismo criterio de selección para ver los resultados.
Una consulta de datos añadidos copia los registros de una o más tablas a otra. Las tablas que contienen los registros que añade no se ven afectadas por la consulta de datos añadidos.
En lugar de añadir registros existentes de otra tabla, puede especificar el valor de cada campo en un registro nuevo utilizando la cláusula VALUES. Si omite la lista de campos, la cláusula VALUES debe incluir un valor para cada campo de la tabla o; de lo contrario; la instrucción INSERT fallará. Utilice una instrucción INSERT INTO adicional con una cláusula VALUES para cada registro adicional que desee crear.

INSERT INTO (Ejemplo de la instrucción)

Este ejemplo selecciona todos los registros de una tabla Nuevos Clientes y los agrega a la tabla Clientes. Cuando no se designan columnas individuales, la nombres de columna de la tabla SELECT deben coincidir exactamente con los de la tabla INSERT INTO.

```
Sub InsertIntoX1()
        Dim dbs As Database
        ' Modifique esta línea para incluir la ruta de
        ' acceso a la base de datos Neptuno en su equipo.
        Set dbs = OpenDatabase("Neptuno.mdb")
        ' selecciona todos los registros de la tabla Nuevos
        ' Clientes y los agrega a la tabla Clientes.
        dbs.Execute " INSERT INTO Clientes " _
                & "SELECT * " _
                & "FROM [Nuevos Clientes];"
        dbs.Close
End Sub
```

Este ejemplo crea un registro nuevo en la tabla Empleados.

```
Sub InsertIntoX2()
        Dim dbs As Database
        ' Modifique esta línea para incluir la ruta de
        ' acceso a la base de datos Neptuno en su equipo.
        Set dbs = OpenDatabase("Neptuno.mdb")
        ' Crea un registro nuevo en la tabla Empleados. El
        ' nombre es Juan, los apellidos Pérez García y el
        ' cargo es Aprendiz.
        dbs.Execute " INSERT INTO Empleados " _
                & "(Nombre,Apellidos,Cargo) VALUES " _
                & "('Juan', 'Pérez García', 'Aprendiz');"
        dbs.Close
End Sub
```

ORDER BY (Cláusula)

Ordena los registros resultantes de una consulta por un campo o unos campos especificados en sentido ascendente o descendente.

Sintaxis:

```
SELECT ListaCampos
FROM Tabla
WHERE CriterioSelección
[ORDER BY campo1 [ASC | DESC ][, campo2 [ASC | DESC ]][, ...]]]
```

Una instrucción SELECT que contenga una cláusula ORDER BY consta de las siguientes partes:

Apartado	Descripción

Listacampos
El nombre del campo o de los campos que se van a recuperar junto con cualquier alias de nombre de campo, funciones de dominio agregado de SQL , predicado de selección (ALL, DISTINCT, DISTINCTROW o TOP) u otras opciones de la instrucción SELECT.

Tabla
El nombre de la tabla o tablas de las que se van a recuperar los datos. Para obtener más información al respecto, vea la cláusula FROM.

CriterioSelección
Criterio de selección. Si la instrucción incluye una cláusula WHERE, el motor de base de datos Microsoft Jet ordena los valores después de aplicar las condiciones de la cláusula WHERE a los registros.

campo1, campo2
Los nombres de los campos por los que se va a ordenar.

Comentarios:

La cláusula ORDER BY es opcional. Sin embargo, si desea que su datos se muestren ordenados, debe utilizar la cláusula ORDER BY. El orden predeterminado es ascendente (A a Z, 0 a 9). Los dos ejemplos siguientes ordenan los nombres de los empleados por su apellido:

```
SELECT Apellidos, Nombre
FROM Empleados
ORDER BY Apellidos;
```

```
SELECT Apellidos, Nombre
FROM Empleados
ORDER BY Apellidos ASC;
```

Para ordenar en sentido descendente (de Z a A,y de 9 a 0), agregue la palabra reservada DESC al final de cada campo por el que desee ordenar en sentido descendente. El siguiente ejemplo selecciona los sueldos y los ordena en sentido descendente:

```
SELECT Apellidos, Salario
FROM Empleados
ORDER BY Salario DESC, Apellidos;
```

Si especifica un campo que contiene datos de tipo Memo u Objeto OLE en la cláusula ORDER BY, se produce un error. El motor de base de datos Microsoft Jet no ordena campos de estos tipos.
La cláusula ORDER BY suele ser el último elemento de una instrucción SQL.
Puede incluir campos adicionales en la cláusula ORDER BY. Los registros se ordenan primero por el primer campo enumerado después de la cláusula ORDER BY. Los registros que tengan valores iguales en ese campo se ordenan entonces por el valor del segundo campo, y así sucesivamente.

ORDER BY (Ejemplo de la cláusula)

La instrucción SQL mostrada en el siguiente ejemplo utiliza la cláusula ORDER BY para ordenar registros por los apellidos en orden descendente (Z-A).

Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub OrderByX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Selecciona los valores del nombre y los
    ' apellidos de la tabla Empleados y los ordena
    ' en orden descendente.
    Set rst = dbs.OpenRecordset("SELECT Apellidos, " _
        & "Nombre FROM Empleados " _
        & "ORDER BY Apellidos DESC;")
    ' Llena el Recordset.
     rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido.
    EnumCampos rst, 12
    dbs.Close
End Sub
```

SELECT...INTO (Instrucción)

Crea una consulta de creación de tabla.
Sintaxis:

SELECT campo1[, campo2[, ...]] INTO nuevatabla [IN basededatosexterna]
FROM origen

La instrucción SELECT...INTO consta de las siguientes partes:

Parte	Descripción
campo1, campo2	El nombre de los campos que se van a copiar a la nueva tabla.
Nuevatabla	El nombre de la tabla que se va a crear. Debe ser conforme con las convenciones de nombres estándar. Si nuevatabla es la misma que el nombre de una tabla ya existente se produce un error interceptable.
Basededatosexterna	La ruta de una base de datos externa. Si desea una descripción de la ruta de acceso, vea la cláusula IN.
Origen	El nombre de la tabla existente de la cual se seleccionan registros. Puede ser una o varias tablas, o una consulta.

Comentarios:

Puede utilizar las consultas de creación de tabla para archivar registros, hacer copias de seguridad de sus tablas o hacer copias para exportar a otra base de datos o utilizar en informes que muestren los datos de un periodo de tiempo concreto. Por ejemplo, podría producir un informe Ventas mensuales por región ejecutando la misma consulta de creación de tabla cada mes.

Notas:

- ✓ Puede que desee definir una clave principal para la nueva tabla. Cuando cree la tabla, los campos de la nueva tabla heredan el tipo de datos y el tamaño de cada campo de las tablas subyacentes de la consulta, pero no se transfiere ninguna otra propiedad del campo o de la tabla.
- ✓ Para agregar datos a una tabla ya existente, utilice la instrucción INSERT INTO en lugar de crear una consulta de datos añadidos.

- ✓ Para ver qué registros se seleccionarán antes de ejecutar la consulta de creación de tabla, primero examine el resultado de una instrucción SELECT que utilice el mismo criterio de selección.

SELECT...INTO (Ejemplo de la instrucción)

Este **ejemplo** selecciona todos los registros de la tabla Empleados y los copia en una nueva tabla llamada Copia Empleados.

```
Sub SelectIntoX()
        Dim dbs As Database
        Dim qdf As QueryDef
        ' Modifique esta línea para incluir la ruta de
        ' acceso a la base de datos Neptuno en su equipo.
        Set dbs = OpenDatabase("Neptuno.mdb")
        ' Selecciona todos los registros de la tabla
        ' Empleados y los copia en una nueva tabla Copia Empleados.
        dbs.Execute "SELECT Empleados.* INTO " _
                & "[Copia Empleados] FROM Empleados;"
        ' Elimina la tabla porque esto es un ejemplo.
dbs.Execute "DROP TABLE [Copia Empleados];"
        dbs.Close
End Sub
```

UNION (Operación)

Crea una consulta de unión, que combina el resultado de dos o más consultas o tablas independientes.
Sintaxis:

[TABLE] consulta1 UNION [ALL] [TABLE] consulta2 [UNION [ALL] [TABLE] consulta n [...]]

La operación UNION consta de las siguientes partes:

Parte	Descripción
Consulta1-n	**Una instrucción SELECT, el nombre de una consulta almacenada o el nombre de una tabla almacenada precedido por la palabra clave TABLE.**

Comentarios:

Puede mezclar el resultado de dos o más consultas, tablas e instrucciones SELECT, en cualquier combinación, en una operación UNION única. El siguiente ejemplo mezcla una tabla existente llamada Nuevas cuentas y una instrucción SELECT:

```
TABLE [Nuevas cuentas] UNION ALL
SELECT *
FROM Clientes
WHERE ImportePedido > 1000;
```

De forma predeterminada, no se devuelven los registros duplicados cuando utiliza una operación UNION; sin embargo, puede incluir el predicado ALL para asegurar que se devuelven todos los registros. Esto también hace que la consulta se ejecute más rápidamente.
Todas las consultas de una operación UNION deber requerir el mismo número de campos. Sin embargo, los campos no tienen que ser del mismo tamaño o tipo de datos.
Utilice alias solamente en la primera cláusula SELECT porque en las demás se ignora. En la cláusula ORDER BY, haga referencia a los campos por cómo se les llama en la primera cláusula SELECT.

Notas:

- ✓ **Puede utilizar la cláusula GROUP BY o HAVING en cada argumento consulta para agrupar los datos devueltos.**
- ✓ **Puede utilizar una cláusula ORDER BY al final del último argumento consulta para mostrar los datos devueltos en un orden específico.**

UNION (Ejemplo de la operación)

Este ejemplo recupera los nombres y las ciudades de todos los proveedores y clientes de España.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub UnionX()
        Dim dbs As Database, rst As Recordset
        ' Modifique esta línea para incluir la ruta de
        ' acceso a la base de datos Neptuno en su equipo.
        Set dbs = OpenDatabase("Neptuno.mdb")
        ' Recupera los nombres y las ciudades de todos los
```

```
        ' proveedores y clientes de España.
        Set rst = dbs.OpenRecordset("SELECT Compañía," _
                & "Ciudad FROM Proveedores" _
                & "WHERE País = 'España' UNION" _
                & "SELECT Compañía, Ciudad FROM Clientes" _
                 & "WHERE País = 'España';")
        ' Llena el Recordset.
        rst.MoveLast
        ' Llama a EnumCampos para imprimir el contenido del
        ' Recordset Transfiere el objeto Recordset y el
        ' tamaño de campo deseado.
        EnumCampos rst, 12
        dbs.Close
End Sub
```

UPDATE (Instrucción)

Crea una consulta de actualización que cambia los valores de los campos en una tabla específica según un criterio específico.

Sintaxis:

```
UPDATE  Tabla  SET NuevoValor   WHERE criterio;
```

La instrucción UPDATE consta de las siguientes partes:

Parte	Descripción

Tabla
El nombre de la tabla cuyos datos desea modificar.

Nuevovalor
Una expresión que determina el valor que se va insertar en un campo concreto de los registros actualizados.

Criterio
Una expresión que determina qué registros se actualizarán. Sólo se actualizan los registros que satisfacen la expresión.

Comentarios:

La instrucción UPDATE es especialmente útil desea cambiar muchos registros o cuando los registros que desea cambiar están en múltiples tablas.
Puede cambiar varios campos al mismo tiempo. El ejemplo siguiente aumenta los valores de Importe Pedido en un 10 por ciento y los valores de Cargo en un 3 por ciento para las compañías de envío de México:

```
UPDATE Pedidos
SET ImportePedido = ImportePedido * 1.1,
Cargo = Cargo * 1.03
WHERE PaísDestinatario = 'México';
```

Importante:

- ✓ La instrucción UPDATE no genera un conjunto de resultado. Además, después de actualizar registros utilizando una consulta de actualización, no puede deshacer la operación. Si desea saber qué registros se actualizaron, examine primero el resultado de una consulta de selección que utilice el mismo criterio y después ejecute la consulta de actualización.
- ✓ Mantenga siempre copias de seguridad de sus datos. Si actualiza registros equivocados, puede recuperarlos desde sus copias de seguridad.

UPDATE (Ejemplo de la instrucción)

Este ejemplo cambia el valor del campo Enumerar de 5 de todos aquellos registros de empleados que actualmente tienen Enumerar con valor 2.

```
Sub UpdateX()
        Dim dbs As Database
        Dim qdf As QueryDef
        ' Modifique esta línea para incluir la ruta de
        ' acceso a la base de datos Neptuno en su equipo.
        Set dbs = OpenDatabase("Neptuno.mdb")
```

```
        ' Cambia el valor del campo Enumerar de 5 de todos
        ' aquellos registros de empleados que actualmente
        ' tienen Enumerar con valor 2.
        dbs.Execute "UPDATE Empleados " _
                & "SET Enumerar = 5 " _
                & "WHERE Enumerar = 2;"
        dbs.Close
End Sub
```

WHERE (Cláusula)

Especifica qué registros de las tablas enumeradas en la cláusula FROM se ven afectados por la instrucción SELECT, UPDATE o DELETE.

Sintaxis:

SELECT Listacampos FROM expresióntabla WHERE criterio

Un instrucción SELECT que contiene una cláusula WHERE consta de las siguientes partes:

Parte	Descripción
Listacampos	El nombre del campo o de los campos que se van a recuperar con cualquier alias de nombre de campo, predicado de selección (ALL, DISTINCT, DISTINCTROW o TOP) u otras opciones de la instrucción SELECT.
Expresióntabla	El nombre de la tabla o de las tablas de las que se van a recuperar los datos.
Criterio	Una expresión que deben cumplir los registros para que se incluyan en el resultado de la consulta.

Comentarios:

El motor de base de datos Microsoft Jet selecciona los registros que cumplen las condiciones enumeradas en la cláusula WHERE. Si no especifica una cláusula WHERE, la consulta devuelve todas las filas de la tabla. Si especifica más de una tabla en la consulta y no ha incluido una cláusula WHERE o una cláusula JOIN, la consulta genera un producto cartesiano de las tablas.

La cláusula WHERE es opcional, pero cuando se incluye, sigue a la cláusula FROM. Por ejemplo, puede seleccionar todos los empleados del departamento de ventas (WHERE Dept = 'Ventas') o todos los clientes que tengan entre 18 y 30 años (WHERE Edad Between 18 And 30).

Si no utiliza una cláusula JOIN para realizar operaciones de combinación SQL en múltiples tablas, el objeto Recordset resultante no podrá actualizar.

La cláusula WHERE es similar a la cláusula HAVING. La cláusula WHERE determina qué registros se seleccionan. De forma parecida, una vez que los registros se agrupan con la cláusula GROUP BY, la cláusula HAVING determina qué registros se van a mostrar.

Utilice la cláusula WHERE para eliminar registros que no desea que se agrupen mediante una cláusula GROUP BY.

Utilice varias expresiones para determinar qué registros devuelve la instrucción SQL. **Por ejemplo**, la siguiente instrucción SQL selecciona todos los empleados cuyos salarios son superiores a $21,000:

```
SELECT Apellidos, Salario
FROM Empleados
WHERE Salario > 21000;
```

Una cláusula WHERE puede contener hasta 40 expresiones vinculadas por operadores lógicos, como And y Or.

Cuando introduce un nombre de campo que contiene un espacio en blanco o un signo de puntuación, escriba el nombre entre corchetes ([ ]). Por ejemplo, una tabla de información de clientes podría incluir información acerca de unos clientes concretos:

```
SELECT [Restaurante favorito del cliente]
```

Cuando especifica el argumento criterio, los literales de fecha deben estar en el formato de EE.UU., incluso si no está utilizando la versión norteamericana del motor de base de datos Microsoft Jet. Por ejemplo, el 10 de mayo de 1996, se escribe 10/5/96 en España y Latinoamérica, y 5/10/96 en Estados Unidos de América. Asegúrese de escribir los literales de fecha entre signos # como se muestra en los **siguientes ejemplos**.

- ✓ Para encontrar registros con fecha 10 de mayo de 1996 en una base de datos española o latinoamericana, debe utilizar la siguiente instrucción SQL:

```
SELECT *   FROM Pedido   WHERE Fecha de envío = #5/10/96#;
```

- ✓ También puede utilizar la función DateValue, que reconoce las configuraciones internacionales establecidas por Microsoft Windows. Por ejemplo, utilice este código para Estados Unidos de América:

```
SELECT *
FROM Pedidos
WHERE Fecha de envío = DateValue('5/10/96');
```

- ✓ **Y utilice este código para España y Latinoamérica:**

```
SELECT *
FROM Pedidos
WHERE Fecha de envío = DateValue('10/5/96');
```

Nota: Si la columna a la que hace referencia en la cadena de criterio es de tipo GUID, la expresión de criterio utiliza una sintaxis ligeramente diferente:

```
WHERE IdRéplica = {GUID {12345678-90AB-CDEF-1234-567890ABCDEF}}
```

Asegúrese de incluir las llaves anidadas ({}) y los guiones como se muestran.

WHERE (Ejemplo de la cláusula)

El siguiente ejemplo supone la existencia de un campo Salario hipotético en la tabla Empleados. Observe que este campo no existe actualmente en la tabla Empleados de la base de datos Neptuno.
Este ejemplo selecciona los campos Apellidos y Nombre de cada registro cuyo apellido es King.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub WhereX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Selecciona los registros de la tabla Empleados
    ' cuyo apellido es King.
    Set rst = dbs.OpenRecordset("SELECT Apellidos, " _
        & "Nombre FROM Empleados " _
        & "WHERE Apellidos = 'King';")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
    ' Recordset.
    EnumCampos rst, 12
    dbs.Close
End Sub
```

WITH OWNERACCESS OPTION (Declaración)

En un entorno multiusuario con un grupo de trabajo seguro, utilice esta declaración con una consulta para conceder al usuario que la ejecute los mismos permisos que el propietario de la consulta.

Sintaxis:

InstrucciónSql
WITH OWNERACCESS OPTION

Comentarios:

- ✓ La declaración WITH OWNERACCESS OPTION es opcional.
- ✓ El **siguiente ejemplo** permite al usuario ver información sobre el salario (incluso si el usuario no tiene permiso para ver la tabla Nómina), proporcionado por el propietario de la consulta que tenga ese permiso:

```
SELECT Apellidos,
Nombre, Salario
FROM Empleados
ORDER BY Apellidos
WITH OWNERACCESS OPTION;
```

- ✓ Si a un usuario se le impide la creación de una tabla o la agregación de datos a ella, puede utilizar WITH OWNERACCESS OPTION para permitir que el usuario ejecute una consulta de creación de tabla o una consulta de datos añadidos.
- ✓ Si desea exigir la configuración de la seguridad de su grupo de trabajo y los permisos de los usuarios, no incluya la declaración WITH OWNERACCESS OPTION.

- ✓ Esta opción requiere que tenga acceso al archivo System.mdw asociado a la base de datos. Es realmente útil sólo en implementaciones seguras de entornos multiusuario.

INNER JOIN (Operación)

Combina registros de dos tablas siempre que existan valores coincidentes en un campo común.

Sintaxis:

FROM Tabla1 INNER JOIN Tabla2 ON Tabla1.Campo1 OperadorComp Tabla2.Campo2

La operación INNER JOIN consta de las siguientes partes:

Parte	Descripción
tabla1, tabla2	Los nombres de las tablas cuyos registros se van a combinar.
campo1, campo2	Los nombres de los campos que se van a combinar. Si no son numéricos, los campos deben ser del mismo tipo de datos y contener la misma clase de datos, pero no tienen porqué tener el mismo nombre.
OperadorComp	Cualquier operador de comparación relacional: "=," "<," ">," "<=," ">=," o "<>."

Comentarios:

Puede utilizar una operación INNER JOIN en cualquier cláusula FROM. Esta es el tipo de combinación más común. Las combinaciones internas combinan los registros de dos tablas siempre que tengan valores coincidentes en un campo común a ambas tablas. Puede utilizar INNER JOIN con las tablas Departamentos y Empleados para seleccionar todos los empleados de cada departamento. En cambio, para seleccionar todos los departamentos (aunque alguno no tenga empleados asignados) o todos los empleados (aunque alguno no esté asignado a ningún departamento), puede utilizar una operación LEFT JOIN o RIGHT JOIN para crear una combinación externa.

Si intenta combinar campos que contienen datos de tipo Memo o Objeto OLE, se produce un error.

Puede combinar dos campos numéricos cualesquiera de tipos similares. Por ejemplo, puede combinar campos AutoNumber y Long puesto que son tipos similares. Sin embargo, no puede combinar campos de tipo Single y Double.

El **siguiente ejemplo** muestra cómo podría combinar las tablas Categorías y Productos por el campo IdCategoría:

```
SELECT NombreCategoría, NombreProducto
FROM Categorías INNER JOIN Productos
ON Categorías.IdCategoría = Productos.IdCategoría;
```

En el **ejemplo anterior**, IdCategoría es el campo combinado, pero no se incluye en la salida de la consulta porque no está incluido en la instrucción SELECT. Para incluir el campo combinado, incluya el nombre de campo en la instrucción SELECT; en este caso, Categorías. IdCategoría.

También puede vincular varias cláusulas ON en una instrucción JOIN, utilizando la sintaxis siguiente:

```
SELECT Campos
FROM tabla1 INNER JOIN tabla2
ON tabla1.campo1 OperadorComp tabla2.campo1 AND
ON tabla1.campo2 OperadorComp tabla2.campo2) OR
ON tabla1.campo3 OperadorComp tabla2.campo3)];
```

También puede anidar instrucciones JOIN utilizando la sintaxis siguiente:

```
SELECT campos
FROM tabla1 INNER JOIN
(tabla2 INNER JOIN [( ]tabla3
[INNER JOIN [( ]tablax [INNER JOIN ...)]
ON tabla3.campo3 OperadorComp tablaX.campoX)]
ON tabla2.campo2 OperadorComp tabla3.campo3)
ON tabla1.campo1 OperadorComp tabla2.campo2;
```

Una operación LEFT JOIN o RIGHT JOIN se puede anidar dentro de una operación INNER JOIN, pero una operación INNER JOIN no se puede anidar dentro de LEFT JOIN o RIGHT JOIN.

INNER JOIN (Ejemplo de la operación)

Este ejemplo crea dos combinaciones equivalentes: una entre las tablas Detalles de pedidos y Pedidos y otra entre las tablas Pedidos y Empleados. Esto es necesario porque la tabla Empleados no contiene datos de ventas y la tabla Detalles de pedidos no contiene datos de empleados. La consulta produce una lista de empleados y sus ventas totales.

Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub InnerJoinX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Crea una combinación entre las tablas Detalles de
    ' pedidos y Pedidos y otra entre las tablas Pedidos
    ' y Empleados. Obtiene una lista de empleados y sus
    ' ventas totales.
    Set rst = dbs.OpenRecordset("SELECT DISTINCTROW " _
        & "Sum(PrecioUnidad * Cantidad) AS Ventas, " _
        & "(Nombre & Chr(32) & Apellidos) AS Nombre " _
        & "FROM Empleados INNER JOIN(Pedidos " _
        & "INNER JOIN [Detalles de pedidos] " _
        & "ON [Detalles de pedidos].IdPedido = " _
        & "Pedidos.IdPedido ) " _
        & "ON Pedidos.IdEmpleado = " _
        & "Empleados.IdEmpleado " _
        & "GROUP BY (Nombre & Chr(32) & Apellidos);")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
    ' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 20
    dbs.Close
End Sub
```

LEFT JOIN, RIGHT JOIN (Operaciones)

Combinan registros de la tabla de origen cuando se utiliza en cualquier cláusula FROM.

Sintaxis:

```
FROM tabla1 [ LEFT | RIGHT ] JOIN tabla2
ON tabla1.campo1 operadorcomp tabla2.campo2
```

Las operaciones LEFT JOIN y RIGHT JOIN consta de las siguientes partes:

Parte	Descripción
tabla1, tabla2	El nombre de las tablas cuyos registros se van a combinar.
campo1, campo2	Nombres de los campos que se combinan. Los campos deben ser del mismo tipo de datos y contener la misma clase de dato, pero no necesitan tener el mismo nombre.
OperadorComp	Cualquier operador de comparación relacional: "=," "<," ">," "<=," ">=," or "<>."

Comentarios:

Utilice una operación LEFT JOIN para crear una combinación externa por la izquierda. Las combinaciones externas por la izquierda pueden incluir todos los registros de la primera (parte izquierda) de dos tablas, aunque no haya valores coincidentes para los registros de la segunda tabla (parte derecha).

Utilice una operación RIGHT JOIN para crear una combinación externa por la derecha. Las combinaciones externas por la izquierda pueden incluir todos los registros de la segunda (parte derecha) de dos tablas, aunque no haya valores coincidentes para los registros de la primera tabla (parte izquierda).

Por ejemplo, podría utilizar LEFT JOIN con las tablas Departamentos (parte izquierda) y Empleados (parte derecha) para seleccionar todos los departamentos, incluyendo aquellos que no tengan empleados asignados a ellos. Para seleccionar todos los empleados, incluyendo aquellos que no estén asignados a ningún departamento, podría utilizar RIGHT JOIN.

El **siguiente ejemplo** muestra cómo podría combinar las tablas Categorías y Productos mediante el campo IdCategoría. La consulta produce una lista de todas las categorías, incluyendo aquellas que no contienen productos:

```
SELECT NombreCategoría,
NombreProducto
FROM Categorías LEFT JOIN Productos
ON Categorías.IdCategoría = Productos.IdCategoría;
```

En este ejemplo, el campo combinado es IdCategoría pero no se incluye en el resultado de la consulta porque no está incluido en la instrucción SELECT. Para incluir el campo combinado, introduzca el nombre del campo en la instrucción SELECT; en este caso, Categorías.IdCategoría.

Notas:

- ✓ **Para crear una consulta que incluya sólo los registros en los cuales los datos de los campos combinados son los mismos, utilice una operación INNER JOIN.**
- ✓ **Una operación LEFT JOIN o RIGHT JOIN se puede anidar en una INNER JOIN, pero una INNER JOIN no se puede anidar en una LEFT JOIN o RIGHT JOIN. Vea la explicación del anidamiento en el tema dedicado a INNER JOIN para ver cómo se anidan combinaciones dentro de otras combinaciones.**
- ✓ **Puede vincular múltiples cláusulas ON. Vea la explicación de la vinculación de cláusulas en el tema dedicado a INNER JOIN para ver cómo se hace.**
- ✓ **Si intenta combinar campos que contienen datos tipo Memo u Objeto OLE, se produce un error.**

LEFT JOIN, RIGHT JOIN (Ejemplo de las operaciones)

Este ejemplo supone la existencia de unos campos NombreDepartamento y IdDepartamento hipotéticos en la tabla Empleados. Observe que este campo no existe actualmente el la tabla Empleados de la base de datos Neptuno.
Este ejemplo selecciona todos los departamentos, incluyendo aquellos que no tienen empleados.
Este ejemplo llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub LeftRightJoinX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' selecciona todos los departamentos, incluyendo
    ' aquellos que no tienen empleados.
    Set rst = dbs.OpenRecordset _
        ("SELECT [NombreDepartamento], " _
        & "Nombre & Chr(32) & Apellidos AS Name " _
        & "FROM Departments LEFT JOIN Empleados " _
         & "ON Departments.[IdDepartamento] = " _
        & "Empleados.[IdDepartamento] " _
        & "ORDER BY [NombreDepartamento];")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
    ' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 20
    dbs.Close
End Sub
```

TRANSFORM (Instrucción)

Crea una consulta de tabla de referencias cruzadas.

Sintaxis:

```
TRANSFORM FunciónDeDominioAgregado
InstrucciónSelect
PIVOT CampoDinámico [IN (Valor1[, Valor2[, ...]])]
```

La instrucción TRANSFORM consta de las siguientes partes:

Parte	Descripción
Funcióndedominioagregado	Una función de dominio agregado de SQL que actúa sobre los datos seleccionados.
Instrucciónselect	Una instrucción SELECT.
Campodinámico	El campo o la expresión que desea utilizar para crear encabezados de columnas en el conjunto de resultado de la consulta.
Valor1, Valor2	Valores fijos utilizados para crear los encabezados de columnas.

Comentarios:

Cuando resume datos utilizando una consulta de tabla de referencias cruzadas, selecciona valores de los campos o expresiones especificadas como encabezados de columna de forma que puede ver los datos en un formato más compacto que con una consulta de selección.

La instrucción TRANSFORM es opcional, pero cuando se incluye es la primera instrucción en una cadena SQL. Precede a una instrucción SELECT que especifica los campos utilizados como encabezados de fila y una cláusula GROUP BY que especifica la agrupación de filas. Opcionalmente, puede incluir otras cláusulas, como WHERE, que especifiquen selecciones adicionales o criterios de selección. También puede utilizar subconsultas como predicados (específicamente, los de la cláusula WHERE) en una consulta de tabla de referencias cruzadas.

Los valores devueltos en campodinámico se utilizan como cabeceras de columna en el conjunto de resultado de la consulta. Por ejemplo, haciendo dinámicas las cifras de ventas en el mes de la venta en una consulta de tabla de referencias cruzadas podría crear 12 columnas. Puede restringir campodinámico para crear encabezados de valores fijos (valor1, valor2) enumerados en la cláusula opcional IN. También puede incluir valores fijos para los cuales no existan datos para crear columnas adicionales.

TRANSFORM (Ejemplo de la instrucción)

Este ejemplo utiliza la cláusula SQL TRANSFORM para crear una consulta de tabla de referencias cruzadas que muestra el número de pedidos obtenidos por cada empleado para en cada trimestre de 1994. Es necesaria la función SalidaSQLTRANSFORM para ejecutar este procedimiento.

```
Sub TransformX1()
    Dim dbs As Database
    Dim strSQL As String
    Dim qdfTRANSFORM As QueryDef
    strSQL = "PARAMETERS MedAño SHORT; TRANSFORM " _
        & "Count(IdPedido) " _
        & "SELECT Nombre & "" "" & Apellidos AS " _
        & "NomCompleto FROM Empleados INNER JOIN Pedidos " _
        & "ON Empleados.IdEmpleado = " _
        & "Pedidos.IdEmpleado WHERE DatePart " _
        & "(""aaaa"", FechaPedido) = [MedAño] "
        strSQL = strSQL & "GROUP BY Nombre & " _
        & """ "" & Apellidos " _
         & "ORDER BY Nombre & "" "" & Apellidos " _
        & "PIVOT DatePart(""q"", FechaPedido)"
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    Set qdfTRANSFORM = dbs.CreateQueryDef _
        ("", strSQL)
    SalidaSQLTRANSFORM qdfTRANSFORM, 1994
    dbs.Close
End Sub
```

Este ejemplo utiliza la cláusula SQL TRANSFORM para crear a consulta de tabla de referencias cruzadas ligeramente más compleja para mostrar el importe total en dólares de los pedidos obtenidos por cada empleado para en cada trimestre de 1994. Es necesaria la función SalidaSQLTRANSFORM para ejecutar este procedimiento.

```
Sub TransformX2()
    Dim dbs As Database
    Dim strSQL As String
    Dim qdfTRANSFORM As QueryDef
    strSQL = "PARAMETERS MedAño SHORT; TRANSFORM " _
        & "Sum(Subtotal) SELECT Nombre & "" """ _
```

```
                & "& Apellidos AS FullName " _
                & "FROM Empleados INNER JOIN " _
                & "(Pedidos INNER JOIN [Subtotales de pedidos] " _
                & "ON Pedidos.IdPedido = " _
                & "[Subtotales de pedidos].IdPedido) " _
                & "ON Empleados.IdEmpleado = " _
                & "Pedidos.IdEmpleado WHERE DatePart" _
                & "(""yyyy"", FechaPedido) = [MedAño] "
                strSQL = strSQL & "GROUP BY Nombre & "" """ _
                & "& Apellidos " _
                & "ORDER BY Nombre & "" "" & Apellidos " _
                & "PIVOT DatePart(""q"",FechaPedido)"
        ' Modifique esta línea para incluir la ruta de
        ' acceso a la base de datos Neptuno en su equipo.
        Set dbs = OpenDatabase("Neptuno.mdb")
        Set qdfTRANSFORM = dbs.CreateQueryDef _
                ("", strSQL)
        SalidaSQLTRANSFORM qdfTRANSFORM, 1994
dbs.Close
End Sub

Function SalidaSQLTRANSFORM(qdfTemp As QueryDef, _
        intAño As Integer)
        Dim rstTRANSFORM As Recordset
        Dim fldBucle As Field
        Dim booPrimero As Boolean
        qdfTemp.PARAMETERS!MedAño = intAño
        Set rstTRANSFORM = qdfTemp.OpenRecordset()
        Debug.Print qdfTemp.SQL
        Debug.Print
        Debug.Print , , "Trimestre"
        With rstTRANSFORM
                booPrimero = True
                For Each fldBucle In .Fields
                        If booPrimero = True Then
                                Debug.Print fldBucle.Name
                                 Debug.Print , ;
                                booPrimero = False
                        Else
                                Debug.Print , fldBucle.Name;
                        End If
                Next fldBucle
                Debug.Print
                Do While Not .EOF
                        booPrimero = True
                        For Each fldBucle In .Fields
                                If booPrimero = True Then
                                        Debug.Print fldBucle
                                        Debug.Print , ;
                                        booPrimero = False
                                Else
                                        Debug.Print , fldBucle;
                                End If
                        Next fldBucle
                        Debug.Print
                        .MoveNext
                 Loop
        End With
End Function
```

Subconsultas SQL

Una subconsulta es una instrucción SELECT anidada dentro de una instrucción SELECT, SELECT...INTO, INSERT...INTO, DELETE, o UPDATE, o dentro de otra subconsulta.

Sintaxis:

Puede utilizar tres formas de sintaxis para crear una subconsulta:

comparación [ANY | ALL | SOME] (instrucciónsql)
expresión [NOT] IN (instrucciónsql)
[NOT] EXISTS (instrucciónsql)

Una subconsulta consta de las siguientes partes:

Parte	Descripción
Comparación	Una expresión y un operador de comparación que compara la expresión con el resultado de la subconsulta.
Expresión	Una expresión para la que se busca el conjunto de resultado de la subconsulta.
Instrucciónsql	Una instrucción SELECT, que sigue el mismo formato y reglas que cualquier otra instrucción SELECT. Debe estar entre paréntesis.

Comentarios:

Puede utilizar una subconsulta en lugar de una expresión en la lista de campos de una instrucción SELECT, o en una cláusula WHERE o HAVING. En una subconsulta, utilice una instrucción SELECT para proporcionar un conjunto de uno o más valores específicos para evaluar en la expresión de la cláusula WHERE o HAVING.
Utilice los predicados ANY o SOME, que son sinónimos, para recuperar registros de la consulta principal que satisfagan la comparación con otros registros recuperados en la subconsulta. El siguiente ejemplo devuelve todos los productos cuyo precio por unidad es mayor que cualquier producto vendido con un descuento del 25 por ciento o mayor:

```
SELECT * FROM Productos
WHERE PrecioUnidad > ANY
(SELECT PrecioUnidad FROM Detalles de pedidos
WHERE Descuento >= .25);
```

Utilice el predicado ALL para recuperar sólo los registros de la consulta principal que satisfagan la comparación con todos los registros recuperados en la subconsulta. Si cambia ANY a ALL en el ejemplo anterior, la consulta devolvería sólo aquellos productos cuyo precio por unidad fuese mayor que el de todos los productos vendidos con un descuento del 25 por ciento o mayor. Esto es mucho más restrictivo.

Utilice el predicado IN para recuperar sólo aquellos registros de la consulta principal para los cuales algún registro de la subconsulta contenga un valor igual. El siguiente ejemplo devuelve todos los productos con descuento del 25 por ciento o mayor:

```
SELECT * FROM Productos
WHERE IdProducto IN
(SELECT IdProducto FROM Detalles de pedidos
WHERE Descuento >= .25);
```

Además, puede utilizar NOT IN para recuperar solamente aquellos registros de la consulta principal para los que ningún registro de la subconsulta contenga un valor igual.

Utilice el predicado EXISTS (con la palabra reservada opcional NOT) en comparaciones verdadero/falso para determinar si la subconsulta devuelve algún registro.

También puede utilizar alias de los nombres de las tablas en la subconsulta para referirse a las tablas enumeradas en una cláusula FROM fuera de la subconsulta. El siguiente ejemplo devuelve los nombres de los empleados cuyos salarios son iguales o mayores que la media de los salarios de todos los empleados que tienen la misma categoría laboral. A la tabla Empleados se le da el alias "T1":

```
SELECT Apellidos,
Nombre, Cargo, Salario
```

```
FROM Empleados AS T1
WHERE Salario >=
(SELECT Avg(Salario)
FROM Empleados
WHERE T1.Cargo= Empleados.Cargo) Order by Cargo;
```

En el ejemplo anterior, la palabra reservada AS es opcional.
En algunas subconsultas se permiten consultas de tabla de referencias cruzadas, específicamente, como predicados (las de la cláusula WHERE). A las subconsultas de salida (las de la lista SELECT) no se les permite consultas de tabla de referencias cruzadas.

SQL (Ejemplo de las subconsultas)

Este ejemplo enumera el los datos de los contactos de todos los clientes que han solicitado un pedido en el segundo trimestre de 1995. **Este ejemplo** llama al procedimiento EnumCampos, que puede encontrar en el ejemplo de la instrucción SELECT.

```
Sub SubQueryX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Enumera el nombre y el contacto de todos los
    ' clientes han solicitado un pedido en el segundo
    ' trimestre de 1995.
    Set rst = dbs.OpenRecordset("SELECT NombreContacto," _
        & "Compañía, CargoContacto, Teléfono" _
        & "FROM Clientes" _
        & "WHERE [ID de cliente]" _
        & "IN (SELECT [ID de cliente] FROM Pedidos" _
        & "WHERE FechaPedido Between #04/1/95#" _
        & "And #07/1/95#);")
    ' Llena el Recordset.
    rst.MoveLast
    ' Llama a EnumCampos para imprimir el contenido del
    ' Recordset. Transfiere el objeto Recordset y el
    ' tamaño de campo deseado.
    EnumCampos rst, 25
    dbs.Close
End Sub
```

DELETE (Instrucción)

Crea una consulta de eliminación que elimina registros de una o más tablas enumeradas en la cláusula FROM y que cumplen la cláusula WHERE

Sintaxis:

```
DELETE [tabla.*]
FROM tabla
WHERE criterio
```

La instrucción DELETE consta de las siguientes partes:

Parte	Descripción

Tabla
El nombre opcional de la tabla cuyos registros se van a eliminar.

Tabla
El nombre de la tabla cuyos registros se van a eliminar.

Criterio
Una expresión que determina qué registros se van a eliminar.

Comentarios:

- ✓ La instrucción DELETE es especialmente útil cuando desea eliminar muchos registros.
- ✓ Para eliminar una tabla completa de la base de datos, puede utilizar el método Execute con la instrucción DROP. Sin embargo, si elimina la tabla, se pierde la estructura. En cambio, cuando utiliza DELETE sólo se elimina el dato; la estructura de la tabla y todas sus propiedades, como los atributos y los índices de los campos, permanecen intactos.

- ✓ Puede utilizar DELETE para quitar registros de las tablas que están en una relación uno a varios con otras tablas. Las operaciones de eliminación en cascada hacen que los registros de las tablas que están en el lado varios de la relación se eliminen cuando el registro correspondiente del lado uno de la relación se elimina en la consulta. Por ejemplo, en la relación entre las tablas Clientes y Pedidos, la tabla Clientes está en el lado uno y la tabla Pedidos está en el lado varios de la relación. La eliminación de un registro de la tabla Clientes provoca que los registros correspondientes de la tabla Pedidos se eliminen si se especifica la opción de eliminación en cascada.
- ✓ Una consulta de eliminación suprime registros completos, y no sólo los datos de los campos específicos. Si desea eliminar valores de un campo específico, cree una consulta de actualización que cambie los valores a Null.

Importante:

- ✓ Después de quitar registros utilizando una consulta de eliminación, no puede deshacer la operación. Si desea saber qué registros se eliminaron, primero examine el resultado con una consulta de selección que utilice el mismo criterio y después ejecute la consulta de eliminación.
- ✓ Mantenga siempre copias de seguridad de sus datos. Si elimina los registros por equivocación, podrá recuperarlos de sus copias de seguridad.

DELETE (Ejemplo de la instrucción)

Este ejemplo elimina todos los registros de empleados cuyo cargo es Aprendiz. Cuando la cláusula FROM incluye sólo una tabla, no tiene que enumerar el nombre de la misma en la instrucción DELETE.

```
Sub DeleteX()
    Dim dbs As Database, rst As Recordset
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Elimina los registros de empleados cuyo cargo es Aprendiz.
    dbs.Execute "DELETE * FROM " _
        & "Empleados WHERE Cargo = 'Aprendiz';"
    dbs.Close

End Sub
```

CREATE TABLE (Instrucción)

Instrucción DDL (Lenguaje de Definición de Datos) que crea una tabla nueva.

Nota: El motor de base de datos Microsoft Jet no permite el uso de la instrucción CREATE TABLE, o cualquiera de las instrucciones de lenguaje de definición de datos (DDL), con motores de base de datos que no sean Microsoft Jet. En vez de ello utilice los métodos Create de DAO.

Sintaxis:

CREATE TABLE tabla (campo1 tipo [(tamaño)] [NOT NULL] [índice1] [, campo2 tipo [(tamaño)] [NOT NULL] [índice2] [, ...]] [, CONSTRAINT índicemultiplescampos [, ...]])

La instrucción CREATE TABLE consta de las siguientes partes:

Parte	Descripción

Tabla
El nombre de la tabla que se va a crear.

campo1, campo2
El nombre del campo o de los campos que se van a crear en la nueva tabla. Debe crear al menos un campo.

Tipo
El tipo de datos de campo en la tabla nueva.

Tamaño
El tamaño del campo en caracteres (solamente campos de tipo Text o Binary).

índice1, índice2

Una cláusula CONSTRAINT que define el índice de campo único. Vea el tema de la cláusula CONSTRAINT para obtener más información acerca de cómo crear este índice.

Índicemultiplescampos

Una cláusula CONSTRAINT que define un índice de múltiples campos. Vea el tema de la cláusula CONSTRAINT para obtener más información acerca de cómo crear este índice.

Comentarios:

Utilice la instrucción CREATE TABLE para definir una tabla nueva, sus campos y limitaciones de campo. Si se especifica NOT NULL para un campo, se exige que los registros nuevos tengan datos válidos en ese campo.

Una cláusula CONSTRAINT establece varias limitaciones en un campo y se puede utilizar para establecer la clave principal. También puede utilizar la instrucción CREATE INDEX para crear una clave principal o índices adicionales en tablas existentes.

Puede utilizar NOT NULL en un único campo o dentro de una cláusula CONSTRAINT con nombre que se aplique a CONSTRAINT de uno o varios campos con nombre. Sin embargo, puede aplicar la limitación NOT NULL solamente a un campo o cuando se produce un error en tiempo de ejecución.

CREATE TABLE, cláusula CONSTRAINT (Ejemplo de la instrucción)

Este ejemplo crea una tabla nueva llamada EstaTabla con dos campos de tipo Text.

```
Sub CreateTableX1()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
' Crea una tabla con dos campos de texto.
dbs.Execute "CREATE TABLE EstaTabla " _
        & "(Nombre TEXT, Apellidos TEXT);"
    dbs.Close
End Sub
```

Este ejemplo crea una tabla nueva llamada MiTabla con dos campos tipo Text, un campo de tipo Fecha/Hora y un índice único hecho con estos tres campos.

```
Sub CreateTableX2()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Crea una tabla con tres campos y un índice
    ' único hecho con estos tres campos.
    dbs.Execute "CREATE TABLE MiTabla " _
        & "(Nombre TEXT, Apellidos TEXT, " _
        & "FechaNacimiento DATETIME, " _
        & "CONSTRAINT MiTablaRestringida UNIQUE " _
        & "(Nombre, Apellidos, FechaNacimiento));"
dbs.Close
End Sub
```

Este ejemplo crea una nueva tabla con dos campos de tipo Texto y un campo de tipo Entero. El campo NSS es la clave principal.

```
Sub CreateTableX3()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Crea una tabla con tres campos y una
    ' clave principal.
    dbs.Execute "CREATE TABLE NuevaTabla " _
        & "(Nombre TEXT, Apellidos TEXT, " _
        & "NSS INTEGER CONSTRAINT MiCampoRestringido " _
        & "PRIMARY KEY);"
    dbs.Close
End Sub
```

ALTER TABLE (Instrucción)

Modifica el diseño de una taba después de que se haya creado con la instrucción CREATE TABLE.

Nota: El motor de base de datos Microsoft Jet no permite el uso de la instrucción ALTER TABLE, o cualquiera de las instrucciones del lenguaje de definición de datos (DDL), con bases de datos que no sean Microsoft Jet. En vez de ello utilice los métodos Create de DAO.

Sintaxis:

ALTER TABLE tabla {ADD {COLUMN campo tipo [(tamaño)] [NOT NULL] [CONSTRAINT índice] |
CONSTRAINT índicemúltiplescampos} |
DROP {COLUMN campoI CONSTRAINT nombreíndice} }

La instrucción ALTER TABLE consta de las siguientes partes:

Parte	Descripción
Tabla	El nombre de la tabla que se va a modificar.
Campo	El nombre del campo que se va a agregar o eliminar de tabla.
Tipo	El tipo de datos del campo.
Tamaño	El tamaño del campo en caracteres (solamente campos tipo Text y Binary).
Indice	El índice de campo. Para obtener más información acerca de cómo construir este índice, vea el tema de la cláusula CONSTRAINT.
IndiceMúltiplesCampos	La definición de un índice de múltiples campos que se va a agregar a tabla. Para obtener más información acerca de cómo construir esta cláusula, vea el tema de la cláusula CONSTRAINT.
NombreIndice	El nombre del índice de múltiples campos que se va a quitar.

Comentarios:

Utilizando la instrucción ALTER TABLE, puede modificar una tabla existente de diferentes formas. Puede:

- ✓ Utilizar ADD COLUMN para agregar un campo nuevo a la tabla. Especifique el nombre del campo, el tipo de datos y un tamaño opcional (para campos tipo Text y Binary). Por ejemplo, la siguiente instrucción agrega una campo tipo Text de 25 caracteres llamado Notas a la tabla Empleados:

ALTER TABLE Empleados ADD COLUMN Notas TEXT(25)

También puede definir un índice en ese campo. Para obtener más información, vea el tema de la cláusula CONSTRAINT.
Si especifica NOT NULL para un campo, se exige que los registros nuevos tengan datos válidos en ese campo.

- ✓ Utilice la instrucción ADD CONSTRAINT para agregar un índice de múltiples campos. Para obtener más información acerca de los índices de múltiples campos, vea el tema de la cláusula CONSTRAINT.
- ✓ Utilice la instrucción DROP COLUMN para eliminar un campo. Especifique solamente el nombre del campo.
- ✓ Utilice la instrucción DROP CONSTRAINT para eliminar un índice de múltiples campos. Especifique solamente el nombre del índice a continuación de la palabra reservada CONSTRAINT.

Notas:

- ✓ No puede agregar o eliminar más de un campo o índice a la vez.
- ✓ Puede utilizar la instrucción CREATE INDEX para agregar un índice de uno o múltiples campos a una tabla, y puede utilizar la instrucción ALTER TABLE o DROP para eliminar un índice creado con la instrucción ALTER TABLE o CREATE INDEX.
- ✓ Puede utilizar NOT NULL en un campo único o dentro de una cláusula CONSTRAINT con nombre que se aplique a uno o varios campos en una cláusula CONSTRAINT con nombre. Sin embargo, sólo puede aplicar la limitación NOT NULL a un campo o cuando se produce un error en tiempo de ejecución.

ALTER TABLE (Ejemplo de la instrucción)

Este ejemplo agrega un campo Salario con un tipo de datos Currency a la tabla Empleados.

```
Sub AlterTableX1()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Agrega el campo Salario a la tabla Empleados
    ' y lo hace de tipo de datos Currency.
    dbs.Execute "ALTER TABLE Empleados " _
        & "ADD COLUMN Salario CURRENCY;"
    dbs.Close
End Sub
```

Este ejemplo quita el campo Salario de la tabla Empleados.

```
Sub AlterTableX2()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' elimina el campo Salario de la tabla Empleados.
    dbs.Execute "ALTER TABLE Empleados " _
        & "DROP COLUMN Salario;"
    dbs.Close
End Sub
```

Este ejemplo agrega una clave externa a la tabla Pedidos. La clave externa se basa en el campo ID de empleado y se refiere al campo ID de empleado de la tabla Empleados. En este ejemplo, no puede enumerar el campo ID de empleado después de la tabla Empleados en la cláusula REFERENCES porque ID de empleado es la clave principal de la tabla Empleados.

```
Sub AlterTableX3()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Agrega una clave externa a la tabla Pedidos.
    dbs.Execute "ALTER TABLE Pedidos " _
        & "ADD CONSTRAINT RelaciónDePedidos " _
        & "FOREIGN KEY ([ID de empleado]) " _
        & "REFERENCES Empleados ([ID de empleado]);"
    dbs.Close
End Sub
```

CONSTRAINT (Cláusula)

Una limitación es similar a un índice, aunque también se puede utilizar para establecer una relación con otra tabla. Puede utilizar la cláusula CONSTRAINT en las instrucciones ALTER TABLE y CREATE TABLE para crear o eliminar limitaciones. Hay dos tipos de cláusulas CONSTRAINT: una para crear una limitación en un único campo y otra para crear una limitación en más de un campo.

Nota: El motor de base de datos Microsoft Jet no permite el uso de la cláusula CONSTRAINT, o cualquier instrucción del lenguaje de definición de datos (DDL), con base de datos que no sean de Microsoft Jet. En vez de ello utilice los métodos Create de DAO.

Sintaxis:

Limitación de un único campo:

```
CONSTRAINT nombre {PRIMARY KEY | UNIQUE | NOT NULL |
REFERENCES tablaexterna [(campoexterno1, campoexterno2)]}
```

Limitación de múltiples campos:

```
CONSTRAINT nombre
{PRIMARY KEY (principal1[,principal2 [, ...]]) |
UNIQUE (único1[,único2 [, ...]]) |
NOT NULL (negadonulo1 [, negadonulo2 [, ...]]) |
FOREIGN KEY (referencia1 [, referencia2 [, ...]]) REFERENCES tablaexterna [(campoexterno1 [,campoexterno2
[, ...]])]}
```

La cláusula CONSTRAINT consta de las siguientes partes:

Parte	Descripción

Nombre
El nombre de la limitación que se va a crear.

principal1, principal2
El nombre del campo o de los campos que se van a designar como clave principal.

único1, único2
El nombre del campo o de los campos que se van a designar como clave única.

negadonulo1, negadonulo2
El nombre del campo o de los campos que se van a limitar a valores no Null.

Referencia1, Referencia2
El nombre del campo o de los campos de la clave externa a los que se hace referencia en otra tabla.

Tablaexterna
El nombre de la tabla externa que contiene el campo o los campos especificados por campoexterno.

campoexterno1, campoexterno2
El nombre del campo o de los campos en tablaexterna especificados por referencia1, referencia2. Puede omitir esta cláusula si el campo al que se hace referencia es la clave principal de tablaexterna.

Comentarios:

Puede utilizar la sintaxis de una limitación de un único campo en la cláusula de definición de campo de una instrucción ALTER TABLE o CREATE TABLE inmediatamente a continuación de la especificación de tipo de datos del campo.
Utilice la sintaxis de limitación de múltiples campos siempre que utilice la palabra reservada CONSTRAINT fuera de la cláusula de definición de campo en una instrucción ALTER TABLE o CREATE TABLE.
Utilizando CONSTRAINT, puede designar un campo como uno de los siguientes tipos de limitaciones:

- ✓ Puede utilizar la palabra reservada UNIQUE para designar un campo como una clave única. Esto quiere decir que no puede haber dos registros en la tabla con el mismo valor en este campo. Puede limitar cualquier campo o lista de campos como único. Si se designa un limitación de múltiples campos como una clave única, los valores combinados de todos los campos del índice deben ser únicos, incluso si dos o más registros tienen el mismo valor en uno de los campos.
- ✓ Puede utilizar las palabras reservadas PRIMARY KEY para designar como clave principal un campo o un conjunto de campos de una tabla. Todos los valores de la clave principal deben ser únicos y no Null, y solamente puede haber una clave principal para una tabla.

Nota: No establezca una limitación PRIMARY KEY en una tabla que ya tenga una clave principal; si lo hace, se produce un error.

Puede utilizar las palabras reservadas FOREIGN KEY para designar un campo como clave externa. Si la clave principal de la tabla externa contiene más de un campo, debe utilizar una definición de limitación de múltiples campos, enumerando todos los campos a los que hace referencia, el nombre de la tabla externa y los nombres de los campos a los que se hace referencia en la tabla externa en el mismo orden en el que se enumeran los campos a los que hace referencia. Si el campo o los campos a los que se hace referencia son la clave principal de la tabla externa, no tiene que especificar los campos a los que se hace referencia; de forma predeterminada, el motor de la base de datos se comporta como si la clave principal de la tabla externa son los campos a los que se hace referencia.

CREATE TABLE, cláusula CONSTRAINT (Ejemplo de la instrucción)

Este ejemplo crea una tabla nueva llamada EstaTabla con dos campos de tipo Text.

```
Sub CreateTableX1()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")

    ' Crea una tabla con dos campos de texto.
     dbs.Execute "CREATE TABLE EstaTabla " _
     & "(Nombre TEXT, Apellidos TEXT);"
    dbs.Close
End Sub
```

Este ejemplo crea una tabla nueva llamada MiTabla con dos campos tipo Text, un campo de tipo Fecha/Hora y un índice único hecho con estos tres campos.

```
Sub CreateTableX2()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
```

```
        Set dbs = OpenDatabase("Neptuno.mdb")
        ' Crea una tabla con tres campos y un índice
        ' único hecho con estos tres campos.
        dbs.Execute "CREATE TABLE MiTabla " _
            & "(Nombre TEXT, Apellidos TEXT, " _
            & "FechaNacimiento DATETIME, " _
            & "CONSTRAINT MiTablaRestringida UNIQUE " _
            & "(Nombre, Apellidos, FechaNacimiento));"
        dbs.Close
    End Sub
```

Este ejemplo crea una nueva tabla con dos campos de tipo Texto y un campo de tipo Entero. El campo NSS es la clave principal.

```
    Sub CreateTableX3()
        Dim dbs As Database
        ' Modifique esta línea para incluir la ruta de
        ' acceso a la base de datos Neptuno en su equipo.
        Set dbs = OpenDatabase("Neptuno.mdb")
        ' Crea una tabla con tres campos y una
        ' clave principal.
        dbs.Execute "CREATE TABLE NuevaTabla " _
            & "(Nombre TEXT, Apellidos TEXT, " _
            & "NSS INTEGER CONSTRAINT MiCampoRestringido " _
            & "PRIMARY KEY);"
        dbs.Close
    End Sub
```

CREATE INDEX (Instrucción)

Crea un índice nuevo en una tabla existente.

Nota: Para bases de datos que no sean de Microsoft Jet, el motor de base de datos Microsoft Jet no permite el uso de la instrucción CREATE INDEX (excepto para crear un pseudo índice en una tabla vinculada de ODBC) o cualquiera de las instrucciones de lenguaje de definición de datos (DDL). En vez de ello, utilice los métodos Create de DAO. Para obtener más información, vea la sección Comentarios.

Sintaxis:

```
CREATE [ UNIQUE ] INDEX índice
ON tabla(campo [ASC|DESC][, campo [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]
```

La instrucción CREATE INDEX consta de las siguientes partes:

Parte	Descripción
Índice	El nombre del índice que se va a crear.
Tabla	El nombre de la tabla existente que contendrá el índice.
Campo	El nombre del campo o de los campos que se van a indexar. Para crear un índice de único campo, escriba el nombre del campo entre paréntesis a continuación del nombre de la tabla. Para crear un índice de múltiples campos, enumere el nombre de cada campo que se va a incluir en el índice. Para crear índices descendentes, utilice la palabra reservada DESC; de lo contrario, se supone que los índices son ascendentes.

Comentarios:

Para no permitir valores duplicados en el campo o campos de índice de diferentes registros, utilice la palabra reservada UNIQUE. En la cláusula opcional WITH puede hacer cumplir reglas de validación. Puede:

- ✓ No permitir entradas Null en el campo o campos de índice de los registros nuevos utilizando la opción DISALLOW NULL.
- ✓ Impedir que se incluyan registros con valores Null en el campo o los campos de índice utilizando la opción IGNORE NULL.
- ✓ Designar el campo o los campos de índice como la clave principal utilizando la palabra reservada PRIMARY. Esto significa que la clave es única, por lo que puede omitir la palabra reservada UNIQUE.

Puede utilizar CREATE INDEX para crear un pseudo índice en una tabla vinculada en un origen de datos ODBC, como SQL Server, que no tenga todavía un índice. No necesita permiso ni tener acceso al servidor remoto para crear un pseudo índice, y la base de datos remota ignora y no le afecta el pseudo índice. Utilice la misma sintaxis para las tablas vinculadas y nativas. Esto se puede especialmente útil para crear un índice en una tabla que sea habitualmente de sólo lectura debido a la carencia de un índice.

También puede utilizar la instrucción ALTER TABLE para agregar un índice de uno o varios campos a una tabla y puede utilizar la instrucción ALTER TABLE o la instrucción DROP para quitar un índice creado con la instrucción ALTER TABLE o CREATE INDEX.

Nota No utilice la palabra reservada PRIMARY cuando cree un nuevo índice en una tabla que ya tenga una clave principal; si lo hace, se produce un error.

CREATE INDEX (Ejemplo de la instrucción)

Este ejemplo crea un índice con los campos Tel domicilio y Extensión tel de la tabla Empleados.

```
Sub CreateIndexX1()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Crea el índice ÍndNue en la tabla Empleados.
    dbs.Execute "CREATE INDEX ÍndNue ON Empleados " _
        & "([Tel domicilio], [Extensión tel]);"
    dbs.Close
End Sub
```

Este ejemplo crea un índice en la tabla Clientes utilizando el campo ID de cliente. Ninguno de los dos registros pueden tener los mismos datos en el campos ID de cliente y no se permiten valores Null.

```
Sub CreateIndexX2()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Crea un índice único, IdClie, con el campo
    ' ID de cliente.
    dbs.Execute "CREATE UNIQUE INDEX IdClie " _
        & "ON Clientes ([ID de cliente]) " _
        & "WITH DISALLOW NULL;"
    dbs.Close
End Sub
```

DROP (Instrucción)

Elimina una tabla existente de una base de datos o elimina un índice existente de una tabla.

Nota: El motor de base de datos Microsoft Jet no permite el uso de la instrucción DROP, o cualquiera de las instrucciones del lenguaje de definición de datos (DDL), con bases de datos que no sean de Microsoft Jet. En vez de ello utilice el método Delete de DAO.

Sintaxis:

DROP {TABLE tabla | INDEX índice ON tabla}

La instrucción DROP consta de las siguientes partes:

Parte	Descripción
Tabla	El nombre de la tabla que se va a eliminar o la tabla de la cual se va a eliminar un índice.
Índice	El nombre del índice que se va a eliminar de tabla.

Comentarios:

Debe cerrar la tabla antes de eliminarla o quitar el índice de ella.
También puede utilizar la instrucción ALTER TABLE para eliminar un índice de una tabla.
Puede utilizar la instrucción CREATE TABLE para crear una tabla y las instrucciones CREATE INDEX o ALTER TABLE para crear un índice. Para modificar una tabla, utilice la instrucción ALTER TABLE.

DROP (Ejemplo de la instrucción)

El siguiente ejemplo supone la existencia de un índice ÍndNue hipotético en la tabla Empleados en la base de datos Neptuno.
Este ejemplo elimina el índice MiÍndice de la tabla Empleados.

```
Sub DropX1()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Elimina ÍndNue de la tabla Empleados.
    dbs.Execute "DROP INDEX ÍndNue ON Empleados;"
    dbs.Close
End Sub
```

Este ejemplo elimina la tabla Empleados de la base de datos.

```
Sub DropX2()
    Dim dbs As Database
    ' Modifique esta línea para incluir la ruta de
    ' acceso a la base de datos Neptuno en su equipo.
    Set dbs = OpenDatabase("Neptuno.mdb")
    ' Elimina la tabla Empleados.
    dbs.Execute "DROP TABLE Empleados;"
    dbs.Close
End Sub
```