Cursos SQL Server 2008 R2

Transact Básico

Antes de empezar

Recomendaciones

- Antes de escribir la consulta se debe decidir qué tablas son necesarias para la consulta.
- Se debe asegurar que la consulta utilice el mínimo de tablas que sea posible. Ya que la unión de tablas adicionales puede hacer que baje el rendimiento.
- Evitar hacer consultas *monolíticas.* Se deben realizar consultas que puedan hacer diferentes cosas desde diferentes partes de la aplicación o devolver datos adicionales por si en un futuro son necesarias.

Determinar qué columnas devolver

Es importante devolver únicamente las columnas necesarias. Si no es así, puede verse afectado el rendimiento por dos factores:

- Utilización de la red.
- Uso de índices. El servidor puede utilizar índices no cluster para satisfacer consultas que usan sólo un subconjunto de las columnas de una tabla. Si añadimos más columnas de las necesarias podemos hacer que la consulta no sea cubierta por estos índices y no utilizarlos con la consiguiente pérdida de rendimiento.

Concatenación de cadenas

- Es un operador de una expresión de cadenas que concatena dos o más cadenas de caracteres o binarias, columnas o una combinación de nombres de columna y cadenas en una expresión.
- Todos los elementos de la concatenación deben ser del mismo tipo (alfanumérico).

Usar la concatenación de cadenas

En el siguiente ejemplo se crea una sola columna en el encabezado de columna ***Name*** de varias columnas de caracteres, con el apellido del contacto seguido de una coma, un solo espacio y, a continuación, el nombre del contacto. El conjunto de resultados está en orden alfabético ascendente por el apellido y, a continuación, por el nombre.

Combinar los tipos de datos

En el siguiente ejemplo se utiliza la función CONVERT (se recomienda CAST) para concatenar los tipos de datos date.

```
USE AdventureWorks;
GO
SELECT 'El pedido se abonará el ' + CONVERT(varchar(12), DueDate, 101)
FROM Sales.SalesOrderHeader
WHERE SalesOrderID = 50001
```

Usar la concatenación de varias cadenas

En el siguiente ejemplo se concatenan varias cadenas para formar una cadena larga que muestra el apellido y la primera inicial de los vicepresidentes de Adventure Works Cycles. Se agrega una coma después del apellido y un punto después de la primera inicial.

(SQLQuery3.sql)

```
SELECT (LastName + ',' + SPACE(1) + SUBSTRING(FirstName, 1, 1) + '.') AS
Name, e.Title
FROM Person.Contact AS c JOIN HumanResources.Employee AS e ON c.ContactID =
e.ContactID
WHERE e.Title LIKE 'Vice%'
ORDER BY LastName ASC;
GO
```

Concatenación en la lista SELECT. En este caso realizaremos la consulta sobre la vista del sistema con la condición 'base table‘ para no seleccionar las vistas.

DISTINCT

El operador Distinct quita los duplicados de un conjunto de filas o de una colección de valores. Es decir, elimina las filas duplicadas de los resultados de una instrucción SELECT. Si no se especifica DISTINCT, se devuelven todas las filas, incluidas las duplicadas. Por ejemplo, si selecciona todos los cargos de la tabla empleados aparecerán cargos duplicados.

Con DISTINCT, se puede eliminar los duplicados y ver sólo los cargos de Empleados que sean únicos.

```
USE Northwind
SELECT title
FROM Employees
SELECT DISTINCT title
FROM Employees
GO
```

La palabra clave DISTINCT elimina las filas duplicadas de los resultados de una instrucción SELECT. Si no se especifica DISTINCT, se devuelven todas las filas, incluidas las duplicadas. Por ejemplo, si selecciona todos los Id. de producto de ProductInventory sin DISTINCT, se devolverán 1069 filas.

Con DISTINCT, puede eliminar los duplicados y ver sólo los Id. de producto que sean únicos.

```
USE AdventureWorks;
GO
SELECT DISTINCT ProductID
FROM Production.ProductInventory
```

Con la palabra clave DISTINCT, se considera que los valores NULL son duplicados unos de otros. Cuando se incluye DISTINCT en una instrucción SELECT, sólo se devuelve un valor NULL en los resultados, con independencia del número de valores NULL que se encuentre.

IDENTITYCOL

Se puede usar IDENTITYCOL para referenciar a una columna IDENTITY (no se admitirá en una versión futura).

```
USE Northwind
GO
SELECT shipperid
FROM Shippers;
SELECT IDENTITYCOL
FROM Shippers
GO
```

ÁLIAS DE COLUMNAS

En ocasiones es obligatorio emplear alias de columnas

```
USE Northwind
SELECT productname + ' (' + quantityperunit + ')' as
cantidades_de_producto,
       unitsinstock + unitsonorder units
FROM Products

GO
```

JOIN Básico

El operador lógico Inner Join devuelve todas las filas que cumplen la combinación de la primera entrada (superior) con la segunda entrada (inferior).

```
USE Northwind
SELECT Territories.territorydescription, Region.regiondescription
FROM Territories JOIN Region
ON Territories.regionid = Region.regionid
GO
```

FROM

La cláusula FROM es obligatoria en todas las instrucciones SELECT en las que se estén recuperando datos de tablas o vistas.

Los nombres de las tablas y vistas se pueden sustituir por alias mediante la cláusula AS.

```
USE Northwind
SELECT T.territorydescription, R.regiondescription
FROM Territories T JOIN Region R
ON T.regionid = R.regionid

GO
```

Tipos de combinación:

- INNER JOIN: Combinación interna. Es la habitual
- LEFT OUTER JOIN: Combinación padres con hijos y padres sin hijos
- RIGHT OUTER JOIN: Combinación hijos con padre e hijos sin padre
- CROSS JOIN: Combinación cartesiana

Ejemplo de INNER JOIN de dos o más tablas:

```
USE AdventureWorks;
GO
SELECT Cst.CustomerID, St.Name, Ord.ShipDate, Ord.Freight
FROM AdventureWorks.Sales.Store AS St
JOIN AdventureWorks.Sales.Customer AS Cst
ON St.CustomerID = Cst.CustomerID
JOIN AdventureWorks.Sales.SalesOrderHeader AS Ord
ON Cst.CustomerID = Ord.CustomerID
```

Ejemplo de LEFT OUTER JOIN de dos tablas:

-- Recuperamos información de los productos vendidos y no vendidos

```
USE AdventureWorks;
GO
SELECT PROD.ProductID, PROD.Name, VEN.OrderQty, VEN.UnitPrice
FROM Production.Product PROD
   LEFT OUTER JOIN Sales.SalesOrderDetail VEN
   ON PROD.ProductID = VEN.ProductID
```

Transact-SQL tiene extensiones que admiten la especificación de objetos que no sean tablas o vistas en la cláusula FROM. Estos otros objetos devuelven un conjunto de resultados, o conjunto de filas en términos de OLE DB, que forman una tabla virtual. La instrucción SELECT funciona entonces como si el conjunto de resultados fuera una tabla.

Se puede referenciar en la cláusula FROM a una tabla de otra base de datos. Cuando se trabaja con servidores vinculados se puede acceder a tablas situadas en una base de datos de otro servidor.

```
USE AdventureWorks
SELECT au_fname + ' ' + au_lname AS name
FROM Pubs..Authors

GO
```

WHERE

Define la condición que se debe cumplir para que se devuelvan las filas. No hay límite en cuanto al número de predicados que se puede incluir en una condición de búsqueda.

En este caso seleccionamos Nombre y el Apellido de los empleados de Seattle.

```
USE AdventureWorks
SELECT contacto.lastname, contacto.firstname, empleado.hiredate
FROM HumanResources.Employee as empleado join Person.Contact as contacto
   ON empleado.ContactID = contacto.ContactID
   join HumanResources.EmployeeAddress as direccion
   ON empleado.EmployeeID = direccion.EmployeeID
   JOIN Person.Address personas
   on direccion.AddressID = personas.AddressID
WHERE personas.city = 'seattle'
```

Ejemplos de filtros WHERE

```
USE Northwind
SELECT lastname, firstname
FROM Employees
WHERE lastname LIKE 'b%'
SELECT lastname, firstname, city
FROM Employees
WHERE city NOT IN ('seattle','redmond','tacoma')
```

Ejemplos de filtros WHERE

```
SELECT lastname, firstname, hiredate
FROM Employees
WHERE hiredate BETWEEN '01/01/1993' AND '31/12/1993'
SELECT lastname, firstname, city
FROM Employees
WHERE city <> 'london'
GO
```

En el siguiente ejemplo se buscan filas en las que la columna LargePhotoFileName tenga los caracteres *green_* y se utiliza la opción ESCAPE porque _ es un carácter comodín. Sin especificar la opción ESCAPE, la consulta buscaría los valores de descripción que contuvieran la palabra green seguida de cualquier carácter distinto del carácter _.

```
USE AdventureWorks ;
GO
SELECT * FROM Production.ProductPhoto
WHERE LargePhotoFileName
LIKE '%green_%' ESCAPE 'a' ;
```

En el siguiente ejemplo se utiliza la cláusula WHERE para recuperar la dirección de correo de una empresa que está fuera de los Estados Unidos (US) y en una ciudad cuyo nombre empieza con A.

```
USE AdventureWorks ;
GO
SELECT AddressLine1, AddressLine2, City, PostalCode, CountryRegionCode
FROM Person.Address AS a
   JOIN Person.StateProvince AS s
     ON a.StateProvinceID = s.StateProvinceID
   WHERE CountryRegionCode NOT IN ('US') AND City LIKE 'A%' ;
```

Operadores AND y OR

Ejemplos de operadores AND y OR

```
SELECT lastname, firstname, city
FROM Employees
WHERE lastname LIKE 'b%'
AND city NOT IN ('seattle','redmond','tacoma')

SELECT lastname, firstname, city, hiredate
FROM Employees
WHERE hiredate BETWEEN '01/01/1993' AND '31/12/1993'
OR city <> 'london'
GO
```

Comparar valores DATETIME

AL comparar valores DATETIME hay que tener cuidado ya que almacena tanto la fecha como la hora.

Por tanto, es muy importante utilizar la función CONVERT / CAST

Actualmente se recomienda usar el tipo DATE si no se desea conservar la hora

```
USE adventureWorks
SELECT salesorderid, customerid, SalesPersonID, orderdate
FROM Sales.SalesOrderHeader
WHERE CONVERT(VARCHAR(20),orderdate,102) > '2004.07.04'
```

Comparaciones con NULL

Selecciona de la lista de proveedores los que tienen/no tienen un valor NULL en la columna región.

```
USE Northwind
SELECT companyname, contactname, region
FROM Suppliers
WHERE region IS NOT NULL
SELECT companyname, contactname, region
FROM Suppliers
WHERE region IS NULL
GO
```

Uso de campos de búsqueda opcionales con diferentes consultas

```
USE Northwind
DECLARE @title VARCHAR(60),
@city VARCHAR(30)
SET @title = NULL
SET @city = 'London'
IF @title IS NOT NULL AND @city IS NULL
  SELECT LastName, FirstName, Title, city
  FROM Employees
  WHERE title = @title
IF @title IS NULL AND @city IS NOT NULL
  SELECT lastname, firstname, title, city
  FROM Employees
  WHERE  City=@city
  IF @title IS NOT NULL AND @city IS NOT NULL
  SELECT lastname, firstname, title, city
  FROM Employees
  WHERE city = @city
  AND title = @title

GO
```

Uso de campos de búsqueda opcionales con diferentes consultas

```
/************************************************
EL caso anterior se puede gestionar con una sola consulta utilizando la función ISNULL
La función ISNULL devuelve el segundo argumento si el primer argumento es nulo y si no
es así devuelve el primer argumento
************************************************/
USE Northwind
DECLARE @title VARCHAR(60), @city VARCHAR(30)
SET @title = NULL
SET @city = 'London'
SELECT lastname, firstname, title, city
FROM Employees
WHERE city = ISNULL(@city,city)
AND title = ISNULL(@title,title)
GO
```

Funciones de agregado

- Las funciones de agregado realizan un cálculo sobre un conjunto de valores y devuelven un solo valor.
- Exceptuando la función COUNT, todas las funciones de agregado ignoran los valores NULL.
- Las funciones de agregado se suelen utilizar con la cláusula GROUP BY de la instrucción SELECT.
- Todas las funciones de agregado son deterministas. Esto significa que las funciones de agregado devuelven el mismo valor cada vez que se las llama con un conjunto específico de valores de entrada.

Las funciones de agregado sólo se pueden utilizar como expresiones en:

- ✓ La lista de selección de una instrucción SELECT (en una subconsulta o en la consulta externa).
- ✓ Cláusulas COMPUTE o COMPUTE BY.
- ✓ Cláusulas HAVING.
- Funciones de agregado

Las funciones de agregado pueden utilizar los siguientes tipos de datos:

- Fecha / Hora
- Numéricos
- Strings

Transact-SQL proporciona las siguientes funciones de agregado (existen más):

- AVG ():Devuelve el promedio (media aritmética) de todos los valores no NULL de un conjunto.

- COUNT (): Devuelve el número de elementos no NULL de un conjunto.
- MAX(): Devuelve el valor máximo de un conjunto.
- MIN (): Devuelve el valor mínimo de un conjunto.
- SUM (): Devuelve la suma de todos los valores de un conjunto.

Funciones de agregado - Ejemplos

```
USE Northwind
SELECT AVG(unitsinstock)
FROM Products
SELECT COUNT(*)
FROM Employees
SELECT MAX(unitprice)
FROM Products
SELECT MIN(birthdate)
FROM Employees
SELECT SUM(unitsinstock)
FROM Products
```

Funciones de agregado - Ejemplos

```
USE Northwind
SELECT COUNT(DISTINCT title)
FROM Employees
GO
USE Northwind
SELECT title, COUNT(*)
FROM Employees
GROUP BY title
GO
USE Northwind
SELECT MAX(orderdate), MIN(orderid)
FROM Orders
GO
```

Funciones de agregado - Ejemplos

```
USE Northwind
SELECT country, COUNT(*)
FROM Customers
WHERE country IN ('Spain','Venezuela')
GROUP BY country
USE Northwind
SELECT country, COUNT(*) AS numero_de_clientes
FROM Customers
WHERE country IN ('Spain','Venezuela')
GROUP BY country
USE Northwind
SELECT country, COUNT(*) AS Numero_de_clientes
FROM Customers
GROUP BY country
HAVING COUNT(*) > 5
GO
```

Funciones de agregado - Ejemplos

```
USE Northwind
SELECT country, COUNT(*) AS Numero_de_Clientes
FROM Customers
GROUP BY country
HAVING COUNT(*) > 5
AND COUNT(*) < 10
GO
```

ORDER BY

Especifica el orden utilizado en las columnas devueltas en una instrucción SELECT. La cláusula ORDER BY no es válida en vistas, funciones en línea, tablas derivadas ni subconsultas, salvo que se especifique también TOP.

```
USE Northwind
SELECT companyname, phone
FROM Shippers
ORDER BY companyname
USE Northwind
SELECT lastname, firstname
FROM Employees
ORDER BY lastname ASC, firstname DESC

GO
```

ORDER BY Con clausula TOP

No es una cláusula estándar ANSI. El argumento TOP no puede ser una variable. Si queremos que sea así necesitamos ejecutar una consulta dinámica (EXEC sp_executesql).

```
USE Northwind
SELECT TOP 10 productid, productname, unitprice
FROM Products
ORDER BY unitprice DESC
SELECT TOP 1 PERCENT productid, productname, unitprice
FROM Products
ORDER BY unitprice DESC
GO
```

Antes se utilizaba SET ROWCOUNT. la cláusula TOP es más eficiente ya que ésta es evaluada en el momento de hacer el análisis sintáctico de la consulta y no el momento de la ejecución como es el caso de ROWCOUNT. Otra desventaja es que hay que desactivarla.

```
USE Northwind
SET ROWCOUNT 10
SELECT productid, productname, unitprice
FROM Products
ORDER BY unitprice DESC
SET ROWCOUNT 0
```

Si queremos que salgan los empates utilizaremos la cláusula TIES. Por ejemplo, sacar los seis productos con mayor cantidad en stock.

```
USE Northwind
SELECT TOP 6 WITH TIES productid, productname, unitsinstock
FROM Products
ORDER BY unitsinstock DESC

GO
```

Consultas dinámicas

Existen dos modos de ejecutar consultas dinámicas:

- EXEC
- sp_executesql (la cadena pasada debe ser de tipo Unicode)

```
USE Northwind
DECLARE @tablename VARCHAR(20), @query NVARCHAR(100)
SET @tablename = 'Shippers'
SET @query = N'SELECT * FROM ' + @tablename
-- Usando EXEC
EXEC (@query)
-- Usando sp_executesql
EXEC sp_executesql @query

GO
```

Desventajas de las consultas dinámicas:

- Las instrucciones contenidas en la ejecución se ejecutan dentro de su propio lote; por tanto, estas instrucciones no pueden acceder a variables declaradas en el lote exterior.
- EXEC no aprovecha el plan de ejecución de la consulta si no se parece lo suficiente a consultas previamente ejecutadas. *sp_executesql* sí lo aprovecha.
- Las consultas dinámicas plantean problemas de seguridad ya que el usuario también necesita permisos sobre los objetos a los que la consulta hace referencia. Esto se debe a que el análisis sintáctico de la consulta dinámica no se lleva a cabo hasta que se ejecuta el procedimiento almacenado, y SQL Server debe controlar los permisos para cada objeto al que la consulta hace referencia (ahora se soluciona con EXECUTE AS).

Consultas dinámicas - Ejemplo

```
USE Northwind
DECLARE @query NVARCHAR(100)
SET @query = N'SELECT * ' + CHAR(13)+ 'FROM Shippers'
-- Para visualizar la consulta (contiene un retorno de carro)
SELECT @query
-- Ejecución de la consulta dinámica
EXEC sp_executesql @query

GO
```

Insert

La instrucción INSERT agrega una o más filas nuevas a una tabla. Tratada de forma simplificada, INSERT tiene el siguiente formato:

INSERT [INTO] table_or_view [(column_list)] data_values

```
USE Northwind
INSERT Territories (territoryid,territorydescription,regionid)
VALUES ('77777','Forst Lauderdale',4)
GO
USE Northwind
INSERT Territories VALUES ('88888','Miami',4)
GO
```

Las instrucciones INSERT no especifican valores para los tipos de columnas con una propiedad IDENTITY que genera valores para la misma.

```
USE Northwind
INSERT Shippers (companyname, phone)
VALUES ('Super Fast Shipping','(503) 555-6493')
GO
```

Las instrucciones INSERT no especifican valores para los tipos de columnas con una propiedad IDENTITY que genera valores para la misma. Siempre que no se especifique lo contrario.

```
USE Northwind
SET IDENTITY_INSERT Shippers ON
INSERT Shippers (shipperid,companyname, phone)
    VALUES (20,'ACME Shipping','(503) 555-8888')
SET IDENTITY_INSERT Shippers OFF
GO
SELECT* FROM shippers

GO
```

Insert's (Formatos)

Dos formas distintas de hacer lo mismo:

```
USE Northwind
INSERT Products (productname,supplierid,categoryid,quantityperunit,
reorderlevel,discontinued)
VALUES ('Donut',NULL,NULL,'6 pieces',DEFAULT,DEFAULT)
-- INSERT Products (productname,quantityperunit)
-- VALUES ('Donut','6 pieces')
GO
```

Inserción de valores predeterminados en todas las columnas

```
USE Northwind
INSERT Orders DEFAULT VALUES

GO
```

Insert's masivas

Vamos a utilizar en este ejemplo una tabla temporal

```
USE Northwind
CREATE TABLE #employees_in_wa (
lastname NVARCHAR(40),
firstname NVARCHAR(20)
)
INSERT #employees_in_wa
SELECT lastname,firstname
FROM Employees
WHERE region = 'WA'
SELECT * FROM #employees_in_wa
GO
```

Vamos a ver lo mismo que el caso anterior pero con un procedimiento almacenado

```
USE Northwind
GO
CREATE PROC get_uk_employees
AS
SELECT lastname,firstname
FROM Employees
WHERE country = 'UK'
GO
CREATE TABLE #employees_in_uk (
lastname NVARCHAR(40),
firstname NVARCHAR(20)
)
INSERT #employees_in_uk
  EXEC get_uk_employees
SELECT * FROM #employees_in_uk
GO
```

Delete

La instrucción DELETE quita una o varias filas de una tabla o vista.

A continuación se expone una forma simplificada de la sintaxis de DELETE:

```
INSERT Orders DEFAULT VALUES
SELECT *
from orders
WHERE customerid IS NULL
DELETE Orders
WHERE customerid IS NULL
SELECT *
from orders

WHERE customerid IS NULL
```

Truncate

La instrucción TRUNCATE TABLE es un método rápido y eficiente para eliminar todas las filas de una tabla. Es como una DELETE sin WHERE pero con estas diferencias:

- No registra en el LOG.
- La tabla no puede tener definidas claves externas.
- No puede contener una cláusula WHERE
- Reinicializa la semilla del valor IDENTITY de la tabla (si hubiera una columna de ese tipo)

```
USE Northwind
GO
CREATE TABLE #shippers (
shid int ,
companyname NVARCHAR(20),
phone NVARCHAR(20)
)
INSERT #shippers
SELECT ShipperID,companyname,phone FROM Shippers
GO
SELECT * FROM #shippers
GO
-- Using TRUNCATE to remove all rows from the #shippers table
TRUNCATE TABLE #shippers
SELECT * FROM #shippers
GO
INSERT #shippers
SELECT ShipperID,companyname,phone FROM Shippers
GO
SELECT * FROM #shippers
GO
Delete #Shippers
GO
SELECT * FROM #shippers
GO
INSERT #shippers
SELECT ShipperID,companyname,phone FROM Shippers
GO
SELECT * FROM #shippers
GO
```

UPDATE

- La instrucción UPDATE puede cambiar los valores de filas individuales, grupos de filas o todas las filas de una tabla o vista.
- Una instrucción UPDATE que haga referencia a una tabla o vista sólo puede cambiar los datos de una tabla a la vez.
- Admite la inclusión de la cláusula JOIN
- Pueden asignarse dentro de ella valores a variables

La instrucción UPDATE tiene las siguientes cláusulas principales:

- SET
  Contiene una lista separada por comas de las columnas que deben actualizarse y el nuevo valor de cada columna con el formato *column_name = expression*
- FROM
  Identifica las tablas o vistas que suministran los valores de las expresiones de la cláusula SET, y las condiciones de combinación opcional entre las tablas o vistas de origen.
- WHERE
  Especifica la condición de búsqueda que define las filas de las tablas y vistas de origen que están calificadas para proporcionar valores para las expresiones de la cláusula SET.

UPDATE - Ejemplos

```
USE Northwind
SELECT *
from Shippers
UPDATE Shippers
SET companyname = companyname + ' Express', phone = '(305) 555 8888'
WHERE shipperid = 20

SELECT *
FROM Shippers
USE AdventureWorks;
UPDATE AdventureWorks.Production.Product
SET ListPrice = ListPrice * 1.1
 WHERE ProductModelID = 37;

USE AdventureWorks;
GO
UPDATE Production.Product
SET Color = N'Metallic Red' WHERE Name LIKE 'Road-250%' AND Color = 'Red';
 GO

USE Northwind
SELECT * FROM Products WHEREproductname = 'Chai'
GO
DECLARE @availableunits SMALLINT
UPDATE Products
SET @availableunits = unitsinstock = unitsinstock + 20
WHERE productname = 'Chai'
GO
PRINT @availableunits
GO
```

SELECT INTO

La instrucción SELECT INTO crea una nueva tabla y la llena con el conjunto de resultados de la instrucción SELECT.

SELECT INTO se puede emplear para combinar datos de varias tablas o vistas en una tabla.

```
USE AdventureWorks; GO
SELECT c.FirstName, c.LastName, e.Title, a.AddressLine1, a.City, sp.Name AS
            [State/Province], a.PostalCode
 INTO dbo.EmployeeAddresses
FROM Person.Contact AS c JOIN HumanResources.Employee AS e
ON e.ContactID = c.ContactID JOIN HumanResources.EmployeeAddress AS ea
ON ea.EmployeeID = e.EmployeeID JOIN Person.Address AS a
ON a.AddressID = ea.AddressID JOIN Person.StateProvince as sp
ON sp.StateProvinceID = a.StateProvinceID;
```

Es obligatorio utilizar un álias para las columnas

```
USE Northwind
SELECT firstname + ' ' + lastname AS NombreCompleto, country
        INTO #employeescountry
FROM Employees
ORDER BY fullname
SELECT * FROM #employeescountry
GO
```

Se puede usar para crear tablas permanentes y temporales

```
USE Northwind
GO
SELECT lastname, firstname
        INTO #salesrep_employees
FROM employees
WHERE title = 'sales representative‘
GO
SELECT * FROM #salesrep_employees
GO
```

SELECT INTO - Ejemplos

Generación de números consecutivos

```
USE Northwind
SELECT IDENTITY(INT,1,1) as companyid, companyname
        INTO #italiancompanies
FROM Clientes
WHERE country = 'Italy‘
SELECT * FROM #italiancompanies
GO
```

Más información

C/ Miracruz, 10 (Bº de Gros) 20001 Donostia

Telf.: 943 275819

email: seim@centroseim.com

S.E.I.M.