CURSO TRANSACT SQL

MANUAL TÉCNICO

Introducción

SQL es un lenguaje normalizado utilizado por diferentes motores de bases de datos para realizar operaciones de consulta y modificación de datos.

Transact-SQL es el lenguaje de base de datos admitido por SQL Server. Todas las aplicaciones que se comunican con SQL Server lo hacen enviando instrucciones Transact-SQL al servidor.

Consultas Básicas

Se pueden diferenciar dos tipos de consultas SQL, las consultas de selección y las consultas de Modificación.

Consultas de Selección

Las consultas de selección devuelven información de las bases de datos, en forma de conjunto de registros o filas

Palabras claves utilizadas en una consulta de selección:

- SELECT: para búsquedas de determinados registros de datos

Cláusulas:

- FROM: para especificar la tabla o tablas de las cuales se van a extraer o modificar los datos.
- AS: para asignar un determinado nombre a la columna o tabla a mostrar. Se puede omitir.
- WHERE: para especificar las condiciones que deben cumplir los datos
- ORDER BY: para especificar la ordenación que deben seguir los datos
- GROUP BY, HAVING: para especificar una determinada agrupación de los datos y establecer condiciones sobre ellos.

Operadores:

- AND, OR, NOT: son operadores lógicos que encadenan condiciones de búsqueda y devuelven 'Verdadero' o 'Falso'. AND devuelve verdadero si todas las condiciones son verdad. OR devuelve verdadero si al menos una de las condiciones es verdad. NOT devuelve lo contrario de la condición
- <, >, <>, <=, >=, = son operadores de comparación de datos. < menor, > mayor, <> distinto, <= menor o igual, >= mayor o igual, = igual.
- BETWEEN operador lógico para especificar un rango de valores.
- LIKE operador lógico para comparar con un modelo de datos.
- IN operador lógico para determina si un valor dado coincide con algún valor de una subconsulta o lista de datos.
- DISTINCT: operador para eliminar los valores duplicados en los campos seleccionados

Funciones Agregadas que devuelven un único valor

- AVG: función utilizada para calcular el promedio de los valores de un campo determinado
- COUNT: función utilizada para devolver el número de registros de la selección
- SUM: función utilizada para devolver la suma de todos los valores de un campo determinado
- MAX: función utilizada para devolver el valor más alto de un campo especificado
- MIN: función utilizada para devolver el valor más bajo de un campo especificado

Sintaxis básica:

SELECT * FROM Tabla

Devuelve toda la información almacenada en una determinada tabla de la base de datos.

SELECT campo1, campo2,...,campoN FROM Tabla

Devuelve los valores de los campos o columnas 1, 2, ..., N de una determina tabla.

SELECT campo1 c1, campo2 c2 ,...,campoN cN FROM Tabla

Devuelve los valores de los campos o columnas 1, 2, ..., N de una determinada tabla y renombra las cabeceras de las columnas.

Aplicando condiciones de búsqueda:

SELECT * FROM Tabla WHERE campo OPERADOR valor

SELECT * FROM Tabla WHERE campo OPERADOR valor ORDER BY campo

Ejemplos:

SELECT * FROM Clientes_Datos

SELECT Cliente,Direccion,IdTipo,NumTelefono,RazonSocial FROM Clientes_Datos

SELECT Cliente Nombre, NumTelefono Tlf FROM Clientes_Datos

En estos ejemplos se muestra información de los clientes existentes en la base de datos.

SELECT * FROM Clientes_Datos WHERE IdTipo=0

En este ejemplo se muestran los datos de todos los clientes de tipo 0 o nacional

SELECT Numalbaran,IdEmpresa,SerieAlbaran,FechaAlb,IdCliente,Embalaje,IdEmpleado

FROM Albaranes_Cli_Cab WHERE IdEstado=2 ORDER BY IdCliente ASC

En este ejemplo se muestran datos de todos los albaranes facturados (estado 2) ordenados ascendentemente por el identificador del cliente

SELECT Numalbaran,IdEmpresa,SerieAlbaran,FechaAlb,IdCliente,Embalaje,IdEmpleado

FROM Albaranes_Cli_Cab WHERE IdEstado=2 AND IdCliente='1145' AND FechaAlb>='20060101'

En este ejemplo se muestran determinados datos de los albaranes facturados al cliente con identificador 1145 a partir del año 2006

SELECT * FROM Clientes_Datos_Comerciales WHERE Sector IN ('METAL', 'AUTO')

Datos de clientes pertenecientes a los sectores del automóvil o del metal

```
SELECT  *  FROM  Clientes_Datos WHERE  Cliente LIKE '%taller%'
```

Aquí se muestran clientes cuyo nombre contiene la palabra taller. El uso del carácter especial % con el operador like significa que se puede sustituir por cualquier carácter.

```
SELECT  COUNT(*)  FROM Clientes_Datos
```

Número de clientes de nuestra base de datos

```
SELECT SUM(Unidades) FROM Contratos_Lineas WHERE IdArticulo='MOTOR HID.' AND IdCliente='1145' AND IdEstado=0
```

Número de motores hidráulicos alquilados actualmente (idestado=2) al cliente 1145

```
SELECT  IdMaquina, SUM(Unidades) AS Cantidad FROM Contratos_Lineas
GROUP BY IdMaquina, IdEstado
HAVING  (IdEstado = 0)
```

En esta consulta se muestran la cantidad de máquinas alquiladas actualmente agrupadas por máquina.

```
SELECT Min(Precio_Euro) FROM  Pedidos_Cli_Lineas WHERE IdArticulo=' MOTOR HID'
```

Mínimo precio al que se vendió el motor hidráulico

```
SELECT  DISTINCT  IdCliente FROM Pedidos_Cli_Cabecera WHERE  Fecha BETWEEN '20060101' AND '20060131'
```

Muestra los clientes a los que se le realizaron pedidos en enero del 2006

Consultas de Modificación

Las consultas de modificación no devuelven información, eliminan, insertan o modifican datos en las tablas especificadas.

Palabras claves utilizadas en una consulta de modificación:

DELETE: para el borrado de datos

INSERT: para la inserción de datos

UPDATE: para la modificación de datos

- DELETE: se usa para la eliminación completa de registros o filas de una o más de las tablas. Una vez que se han eliminado los registros requeridos, no puede deshacer la operación.

  Sintaxis básica:
  DELETE FROM Tabla WHERE Condicion

  Ejemplos:

  DELETE FROM Pedidos_cli_lineas WHERE IdEstado =-1

  En este ejemplo se eliminan todas las líneas de pedidos de cliente con estado anulado.

  DELETE FROM Empleados_Datos WHERE FechaAlta <'20000101'

  En este ejemplo se eliminara todos los empleados dados de alta con anterioridad al año 2000. Hay que tener presente que en estas consulta de eliminación si se omite la cláusula WHERE todos los registros de la serán borrados.

- INSERT INTO: se usa para añadir uno o varios registros a una tabla determinada.

  Sintaxis básica:

  INSERT INTO Tabla (campo1, campo2, .., campoN) VALUES (valor1, valor2, ..., valorN)

  Para agregar un solo registro o bien:

INSERT INTO Tabla (campo1, campo2, .., campoN)

SELECT campo1, campo2, ..., campoN FROM Tablaorigen WHERE Condición

Para agregar los registro de una tabla que cumplan determinada condición.

Ejemplos:
INSERT INTO Empleados_Datos (IdEmpleado, Nombre, Comision)

VALUES (22,'Maria',100)

En este ejemplo se agrega un nuevo empleado a la base de datos. Nótese que a pesar de que la tabla tiene muchos campos no es necesario proporcionarlos todos ya que en la mayoría de ellos o bien admiten nulos o bien tienen un valor por defecto asignado

- UPDATE: se usa para la actualización de uno o varios campos de una tabla basándose o no en un determinado criterio. Si en esta consulta de actualización se omite la cláusula WHERE todos los registros de la tabla requerida serán actualizados.

  Sintaxis Básica:

  UPDATE Tabla SET Campo1=Valor1, Campo2=Valor2, ... CampoN=ValorN WHERE Condición

  Ejemplos:

  UPDATE Empleados_Datos SET Salario = Salario*1.1

  En este ejemplo a todos los empleados de nuestra base de datos se les modifica el salario aumentándolo en un 10 por cien.

  UPDATE Listas_Precios_Cli_Art SET Precio=0

  WHERE Idlista=1 AND IdArticulo='ArtRegalo' AND DesdeFecha>'20060101'

  En este ejemplo se modifica el precio a 0 de la lista de precios con identificador 1 para el artículo con descripción ArtRegalo del presente año.

Hay que tener en cuenta que las consultas Delete, Insert y Update pueden generar errores debido a causas muy concretas, lo que impediría que se ejecutase la acción requerida. Las principales causas de error son las siguientes:

- Se intenta eliminar o modificar un registro de una tabla que forma parte de una clave ajena o una regla de negocio programa en el trigger de la tabla.
- Se intenta insertar valores nulos en campos de la tabla que no lo permiten.
- Se intenta insertar registros en campos auto numéricos que se rellenan automáticamente
- Se intenta insertar registros que violan la clave principal o ajena de la tabla provocando registros duplicados o inconsistencia de datos.

Ejemplos:
DELETE FROM Clientes_Sectores WHERE Sector='Metal'
Este ejemplo no produciría ninguna acción de borrado si en nuestra base de datos tenemos clientes que pertenecen al sector del metal ya que existe una clave ajena en la tabla que lo impide.

INSERT INTO Clientes_Sectores (Sector,DescripcionSec) VALUES('Metal', 'Sector del Metal')
Este ejemplo no producirá ninguna acción de inserción si ya tenemos dado de alta en nuestra base de datos el sector del metal.

Es deseable que antes de ejecutar cualquier consulta de modificación sepamos que registros van a resultar modificados examinándolos primero con la consiguiente consulta de selección (utilizando la misma condición).

Consultas de Combinación

Hasta ahora todos los ejemplos mostrados requieren datos de una sola tabla, pero es habitual tener que extraer información de varias tablas a la vez.

Las vinculaciones entre varias tablas que se realizan mediante la cláusula INNER, esta cláusula combina registros de dos tablas siempre que haya concordancia de valores en un campo común.

Sintaxis Básica:

SELECT * FROM tabla1 INNER JOIN tabla2 ON tabla1.campo1 = tabla2.campo2

SELECT * FROM tabla1 LEFT JOIN tabla2 ON tabla1.campo1 = tabla2.campo2

SELECT * FROM tabla1 RIGHT JOIN tabla2 ON tabla1.campo1 = tabla2.campo2

Donde campo1 y campo2 son los nombres de los campos que se combinan. Estos campos deben ser del mismo tipo de datos.

Ejemplos:

```
SELECT  DISTINCT  (Cliente) FROM Clientes_Datos C
INNER JOIN Pedidos_Cli_Cabecera  P ON  C.IdCliente =P.IdCliente
```

En este ejemplo se muestran todos los clientes de nuestra base de datos a los que se ha realizado algún pedido. Obsérvese que omitiendo la palabra DISTINCT un mismo cliente saldría repetido tantas veces como pedidos tuviese asociados.

```
SELECT  DISTINCT  (Cliente) FROM Clientes_Datos C
LEFT JOIN Pedidos_Cli_Cabecera  P ON  C.IdCliente =P.IdCliente
```

En este otro ejemplo se mostrarían todos los clientes, tuviesen o no pedidos asociados

```
SELECT Albaranes_Cli_Cab.IdAlbaran,
        Facturas_Cli_Cab.IdFactura,
        Pedidos_Cli_Lineas.*
FROM    Pedidos_Cli_Lineas
        INNER JOIN Albaranes_Cli_Cab ON
        Pedidos_Cli_Lineas.IdAlbaran = Albaranes_Cli_Cab.IdAlbaran
        LEFT JOIN Facturas_Cli_Cab ON
        Pedidos_Cli_Lineas.IdFactura = Facturas_Cli_Cab.IdFactura
```

En este ejemplo se muestra la información de los pedidos albaraneados facturados y no facturados. Si el pedido no está facturado la columna IdFactura aparecerá rellenada con nulos, por el contrario la columna IdAlbaran siempre tendrá un dato valido ya que se ha combinado con la cláusula INNER en vez de LEFT

Si empleamos la cláusula INNER en una consulta se seleccionarán sólo aquellos registros de la tabla que hayamos escrito a la izquierda de INNER JOIN que contengan al menos un registro de la tabla que hayamos escrito a la derecha. Las cláusulas LEFT y RIGHT actúan de un modo diferente.
LEFT toma todos los registros coincidentes en ambas tablas y también los registros de la tabla de la izquierda aunque no tengan ningún registro coincidente en la tabla derecha. RIGHT realiza la misma operación pero al contrario, toma todos los registros coincidentes en ambas tablas y los registros de la tabla de la derecha aunque no tenga ningún registro en la tabla de la izquierda.

Consultas de Unión

Se utiliza la operación UNION para crear una consulta de unión que combina los resultados de dos o más consultas o tablas independientes. Su sintaxis es:

consulta1

UNION [ALL]

consulta2

UNION [ALL]

ConsultaN

El operador ALL incorpora todas las filas en los resultados, incluidas las duplicadas. Si no se especifica, se quitan las filas duplicadas.
Todas las consultas en una operación UNION deben pedir el mismo número de campos, no obstante los campos no tienen por qué tener el mismo tamaño o el mismo tipo de datos.

En el siguiente ejemplo se muestran las direcciones de clientes y proveedores en Valencia

```
SELECT cliente,ciudad,direccion FROM Clientes_Datos WHERE ciudad='valencia'
UNION
SELECT proveedor,ciudad,direccion FROM Prov_Datos WHERE ciudad='valencia'
```

SELECT LP.IdLista, LPA.IdArticulo, LPA.Precio FROM Listas_Precios_Cli LP INNER JOIN Listas_Precios_Cli_Art LPA ON LP.IdLista = LPA.IdLista

UNION

SELECT IdLista, IdArticulo, Precio FROM Listas_Precios_Cli_HArt

En este ejemplo se muestran datos de las listas de precios activas junto a datos de las listas de precios del histórico.

Subconsultas

Una subconsulta es una instrucción SELECT anidada dentro de otra subconsulta o dentro de una instrucción SELECT, SELECT...INTO, INSERT...INTO, DELETE, o UPDATE.
La consulta SELECT de una subconsulta se incluye siempre entre paréntesis.

La sintaxis más habitual de las subconsulta esta ligada a las cláusulas: IN y EXISTS (con la palabra reservada NOT opcional)

IN se emplea para recuperar únicamente aquellos registros de la consulta principal para los que algunos registros de la subconsulta contienen un valor igual. EXISTS se utiliza para determinar si la subconsulta devuelve algún registro.

Por ejemplo si deseamos saber todos aquellos clientes que hayan realizado al menos un pedido:
SELECT Cliente,NumTelefono FROM Clientes_Datos WHERE EXISTS
(SELECT * FROM Pedidos_Cli_Cabecera WHERE IdCliente = Clientes_Datos.IdCliente)

Este ejemplo sería equivalente a:

```
SELECT Cliente,NumTelefono FROM Clientes_Datos WHERE IdCliente IN
        (SELECT IdCliente FROM Pedidos_Cli_Cabecera)
```

Utilizar Información de otra base de Datos

Si tenemos varias bases de datos en nuestro servidor y deseamos utilizar información de una tabla que no se encuentra en la base de datos en la que se ejecuta la consulta, podemos acceder a ella anteponiendo simplemente el nombre de la base de datos:

```
SELECT  *  FROM  BaseDatos.dbo.Clientes_Datos
```

También es posible acceder a información situada en otro servidor anteponiendo el nombre del servidor remoto. Aunque para esto último los servidores han de estar enlazados

Consultas útiles:

select distinct o.name,o.type from sys.syscomments c inner join sys.sysobjects o on c.id=o.id where text like '%articulos%'

Select o.name,c.name from sys.sysobjects o inner join sys.syscolumns c on o.id=c.id where c.name like '%generic%' and o.xtype='u'

Ejemplos

Trigger Insert

```
ALTER TRIGGER [dbo].[Articulos_Familias_ITrig]
ON [dbo].[Articulos_Familias]
FOR INSERT
AS

Declare @ins int

IF @@ROWCOUNT<>0 BEGIN

	--Comprobar la integridad con LineasNegocio
	SELECT @Ins=COUNT(I.IdLineaNegocio)

	FROM Inserted I LEFT JOIN LineasNegocio LN ON I.IdLineaNegocio= LN.IdLineaNegocio

	WHERE I.IdLineaNegocio IS NOT NULL AND LN.IdLineaNegocio IS NULL

	IF @Ins>0 BEGIN
		PRINT 'Error a insertar Familia de Articulo. La LineaNegocio No Existente.'
		ROLLBACK TRANSACTION
		RETURN
	END
END
```

Vista recursiva:

```
--familias hijas
;WITH FAM AS
(
SELECT IdFamilia as IdFam FROM Articulos_Familias WHERE IdFamilia='prueba1'
  UNION ALL
SELECT AF.IdFamilia FROM FAM F INNER JOIN Articulos_Familias AF ON F.IdFam=AF.IdFamiliaPadre
)
SELECT * FROM FAM
```

Crear auto numéricos:

```
SELECT IdEjercicio,Asiento,Apunte
        ,ROW_NUMBER() OVER (PARTITION BY Asiento ORDER BY Apunte) IdApunte_NEW
        ,ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) Autonomerico
FROM Conta_Apuntes WHERE IdEjercicio = 1 AND Asiento IN (2,4)
```

Último precio compra año:

```
CREATE VIEW Ultimo_Precio_Compra_Articulo_Año
AS

WITH Datos (ID,IdArticulo,Año,FechaFactura,Precio)
AS (SELECT ROW_NUMBER() OVER (PARTITION BY YEAR(f.FechaFact),l.IdArticulo ORDER BY YEAR(f.FechaFact), L.IdArticulo), L.IdArticulo, YEAR(F.FechaFact), L.FechaFactura, L.Precio_EURO
```

```
        FROM Pedidos_Prov_Lineas L INNER JOIN Facturas_Prov_Cab F ON F.IdFactura=L.IdFactura)

,Maximos (ID,IdArticulo,Año)
        AS (SELECT MAX(ID), IdArticulo, Año FROM Datos GROUP BY IdArticulo,Año)

SELECT D.IdArticulo,D.Año,D.FechaFactura,D.Precio
FROM Datos D INNER JOIN Maximos M ON D.ID=M.ID AND D.IdArticulo=M.IdArticulo AND D.Año=M.Año
go

SELECT * FROM Ultimo_Precio_Compra_Articulo_Año
```

Cursor:

```
--Declaración

DECLARE nombre_cursor CURSOR FOR
        sentencia select
OPEN nombre_cursor
FETCH NEXT FROM nombre_cursor INTO       @variable
WHILE @@FETCH_STATUS<>-1 BEGIN

-- siguiente registro
        FETCH NEXT FROM nombre_cursor INTO       @variable
END
CLOSE nombre_cursor
DEALLOCATE nombre_cursor
```

-Ejemplo

```
Declare @name varchar(100)
Declare @Cont int
Declare @tab TABLE (tabla varchar(100), numReg int)

DECLARE Tabs CURSOR FOR
        SELECT name FROM sysobjects WERE type='u' ORDER BY 1
OPEN Tabs
FETCH NEXT FROM tabs INTO @name
WHILE @@FETCH_STATUS<>-1 BEGIN

        INSERT INTO @tab (tabla,numReg)
        EXEC('SELECT ''' + @name+ ''''' + ', COUNT(*) FROM ' + @name)

        -- siguiente
        FETCH NEXT FROM tabs INTO @name
END
CLOSE tabs
DEALLOCATE tabs

SELECT * FROM @tab order by numReg
```

Plantilla Stored Procedure:

```
CREATE PROCEDURE Stored_Prueba
AS
- plantilla stored
BEGIN
```

```
	-- declaraciones
	-- codigo

	BEGIN TRY
	BEGIN TRAN
		-- codigo critico
		IF @@ROWCOUNT=0
			RAISERROR('Error en codigo.',12,1)

	COMMIT TRAN
	END TRY
	BEGIN CATCH
		IF @@TRANCOUNT >0 BEGIN
			ROLLBACK TRAN
		END
		PRINT  'Error'
		-- retorno KO
		RETURN 0
	END CATCH

	-- retorno OK
	RETURN -1
END
```

Procedimiento almacenado:

```
CREATE PROCEDURE [dbo].[PDameImportesFactura]
	@IdFactura T_Id_Factura,
	@Anyo int OUTPUT,
```

```
        @Mes int OUTPUT,
        @PVP Float OUTPUT,
        @Precio_Total Float OUTPUT
AS
-----------------------------------------------------------------------------------------------
-- procedimiento para devolver datos de factura
-----------------------------------------------------------------------------------------------
BEGIN
        -- comprobar si se ha especificado la factura
        IF @IdFactura IS NULL BEGIN
                PRINT 'No se ha especificado la factura.'
                RETURN 0
        END

        -- rellenar variables salida
        SELECT @Anyo=YEAR(F.FechaFact),
                @Mes=MONTH(F.FechaFact),
@PVP=SUM(l.Precio_EURO * (1-L.Descuento/100.0) * (1-F.Descuento/100.0) * (1-F.DescuentoPP/100.0)),
@Precio_Total=SUM(L.Cantidad * L.Precio_EURO * (1-L.Descuento/100.0) * (1-F.Descuento/100.0) * (1-F.DescuentoPP/100.0))
        FROM Pedidos_Cli_Lineas L INNER JOIN Facturas_Cli_Cab F ON L.IdFactura=F.IdFactura
        WHERE F.IdFactura=@IdFactura
        GROUP BY F.IdFactura,F.FechaFact

        -- retorno
        RETURN -1
```

```
END


-- llamada
Declare @Anyo int
Declare @Mes int
Declare @PVP Float
Declare @Precio_Total Float
Declare @ret int

EXEC @re t= PDameImportesFactura 24, @Anyo OUTPUT, @Mes OUTPUT, @PVP OUTPUT, @Precio_Total OUTPUT

PRINT @Anyo
PRINT @Mes
PRINT @PVP
PRINT @Precio_Total
PRINT @ret
```

Funciones escalares:

```
--ejemplo 1

CREATE FUNCTION [dbo].[funEsDiaFestivo_Emp](@IdEmpleado T_Id_Empleado, @Fecha DATETIME)
RETURNS bit
AS

BEGIN
        DECLARE @val bit
```

```
IF EXISTS (SELECT * FROM Calendario_Fechas CF INNER JOIN Empleados_Datos ED ON CF.IdCalendario = ED.IdCalendario WHERE ED.IdEmpleado = @IdEmpleado AND CF.Fecha = @Fecha)

                SET @val=1
        ELSE
                SET @val=0

        RETURN @val
END
go
-- llamada
SELECT  dbo.[funEsDiaFestivo_Emp] (0,'20130105')
go
--ejemplo 2
CREATE FUNCTION [dbo].[funUltDiaMes](@Fecha T_Fecha_Corta)
RETURNS smallint
AS
BEGIN
Declare @PrimDiaMesSig T_Fecha_corta
Declare @UltDiaMes T_Fecha_corta

        SET @PrimDiaMesSig=LEFT(convert(varchar,DATEADD(mm,1,@fecha),112),6)+'01'
        SELECT @UltDiaMes=DATEADD(dd,-1,@PrimDiaMesSig)

        RETURN DATEPART(dd,@UltDiaMes)
END
go
```

```
-- llamada

SELECT  dbo.[ funUltDiaMes] (‘20130105')
```

Funciones tipo tabla:

```
CREATE FUNCTION [dbo].[Fun_DameDatosFactura] (@IdFactura T_Id_Factura=NULL)

 RETURNS @RET TABLE (Anyo int, Mes int,IdFactura int, IdArticulo varchar(50),Cant decimal (38,10),PVP Float, Precio_Total Float)


-- funcion para devolver datos de factura

BEGIN

Declare @Tab_Facts TABLE (IdFactura int)


        -- rellenar @Tab_Facts

        IF @IdFactura IS NOT NULL

                INSERT INTO @Tab_Facts VALUES (@IdFactura)

        ELSE

                INSERT INTO @Tab_Facts (IdFactura)

                SELECT IdFactura FROM Facturas_Cli_Cab


        -- rellenar tabla retorno

        INSERT INTO @RET (Anyo,Mes,IdFactura,IdArticulo,Cantidad,PVP,Precio_Total)

        SELECT YEAR(F.FechaFact),

                MONTH(F.FechaFact),

                F.IdFactura,

                l.IdArticulo,

                l.Cantidad,

l.Precio_EURO * (1-L.Descuento/100.0) * (1-F.Descuento/100.0) * (1-F.DescuentoPP/100.0) PVP,
```

```
L.Cantidad * L.Precio_EURO * (1-L.Descuento/100.0) * (1-F.Descuento/100.0) * (1-F.DescuentoPP/100.0) Precio_Total
        FROM @Tab_Facts T INNER JOIN Pedidos_Cli_Lineas L ON T.IdFactura=L.IdFactura
                INNER JOIN Facturas_Cli_Cab F  WITH (NOLOCK) ON L.IdFactura=F.IdFactura

        -- retorno
        RETURN

END
-- llamada
SELECT * FROM dbo.[Fun_DameDatosFactura](default)

-- llamada

SELECT * FROM Facturas_cli_cab F CROSS APPLY dbo.[Fun_DameDatosFactura](F.IdFactura) l
```

Funciones tipo tabla en línea:

```
CREATE FUNCTION [dbo].[funSeRequiereLotes](@IdArticulo T_Id_Articulo, @IdAlmacen T_Id_Almacen)
RETURNS TABLE AS
RETURN
        SELECT TOP 1 CAST(SUB.L AS BIT) ConLotes
        FROM (
SELECT COUNT(IdAlmacen) L FROM Almacenes WHERE IdAlmacen=@IdAlmacen AND Lotes=1
                UNION
SELECT COUNT(IdArticulo) L FROM Articulos WHERE IdArticulo=@IdArticulo AND Lotes=1
        ) SUB
        ORDER BY SUB.L
```

```
go

--llamada


SELECT * FROM dbo.funSeRequiereLotes (‘pintura’,0)


Go

-- llamada

SELECT L.IdPedido,L.IdLinea,F.ConLotes FROM Pedidos_Cli_Lineas L CROSS APPLY
dbo.funSeRequiereLotes(L.IdArticulo, L.IdAlmacen) F
```

Consultas dinámicas:

```
DECLARE @valor int


EXECUTE sp_executesql N'SELECT @ret = COUNT(*) FROM Pedidos_Cli_Lineas

 WHERE Precio_Euro>@MinPrecio',

N'@MinPrecio int, @ret int OUTPUT',@MinPrecio = 10,@ret=@valor OUTPUT

PRINT @valor
```