Capítulo XIII

Funciones Definidas por el Usuario

En este capítulo aprenderemos a crear nuestras propias funciones. Tal como en otros lenguajes de programación, en Transact-SQL es posible definir funciones, que son subrutinas que pueden recibir parámetros, ejecutar un proceso y entregar el resultado del proceso como un valor de retorno.

En SQL Server podemos diseñar nuestras propias funciones para complementar y extender las funciones estándar incorporadas con el sistema.

1. ¿QUÉ ES UNA FUNCIÓN DEFINIDA POR EL USUARIO?

Una función definida por el usuario es muy similar a un procedimiento almacenado: recibe 0, 1 ó más parámetros de entrada conocidos como argumentos de la función, ejecuta un proceso, y entrega el resultado del proceso como un valor de retorno escalar (un solo valor) ó como un conjunto de valores (una tabla). A diferencia de un procedimiento almacenado, la función definida por el usuario no soporta parámetros de salida; el resultado se devuelve en el valor de retorno de la función.

Los parámetros de entrada pueden ser de cualquier tipo excepto **timestamp**, **cursor** ó **table**.

1.1. Tipos de funciones definidas por el usuario

SQL Server presenta tres tipos de funciones definidas por el usuario:

- Función escalar
- Función tabla evaluada en línea
- Función tabla evaluada multisentencia

2. FUNCIÓN ESCALAR

Es similar a las funciones estándar de SQL Server y entrega como valor de retorno un solo valor del tipo indicado en su cláusula RETURNS.

2.1. Creación de una función escalar- La sentencia CREATE FUNCTION

Sintáxis

```
CREATE FUNCTION nombreFunción(
   @parámetro1 tipo_dato [ = valor ] ,
   @parámetro2 tipo_dato [ = valor ] , ...
   RETURNS tipo_dato_valor_retorno
[ AS ]
BEGIN
   sentenciasSQL
   RETURN valorRetorno
END
```

Ejercicio 13.1: Creación de una función escalar

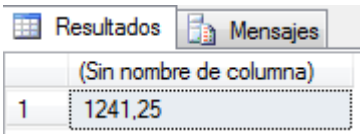
Este ejercicio define una función **udf_MontoGuia** que retorna el monto de la guía de envío número N.

```
USE QhatuPERU
go

CREATE FUNCTION udf_MontoGuia(
     @guia int)
     RETURNS money
BEGIN
     RETURN (SELECT SUM(PrecioVenta *
               CantidadEnviada)
          FROM GUIA_DETALLE
          WHERE NumGuia = @guia)
END
go
```

El siguiente batch invoca a la función **udf_MontoGuia** y obtiene el monto de la guía número **77**.

```
DECLARE @laGuia int
SET @laGuia = 77
SELECT dbo.udf_MontoGuia(@laGuia)
go
```



Observe que para invocar a la función escalar debemos especificar el esquema al que pertenece; en este caso, el esquema predeterminado **dbo**.

3. FUNCIÓN TABLA EVALUADA EN LÍNEA

Es una función que ejecuta una sola sentencia SELECT y retorna el resultado en formato de tabla. Es similar a una vista, pero ofrece más flexibilidad en el uso de parámetros. El tipo del valor de retorno es **table**.

3.1. Creación de una función tabla evaluada en línea

Ejercicio 13.2: Función tabla evaluada en línea

La función **udf_ProveedorLineaArticulo** entrega en una variable de tipo table una lista de los proveedores de los artículos de la línea cuyo código se le pasa.

```
USE QhatuPERU
go

CREATE FUNCTION udf_ProveedorLineaArticulo(
     @linea integer)
     RETURNS table
AS
RETURN (SELECT DISTINCT ARTICULO.CodProveedor,
          PROVEEDOR.NomProveedor,
          PROVEEDOR.Direccion, PROVEEDOR.Ciudad
FROM ARTICULO INNER JOIN PROVEEDOR
     ON ARTICULO.CodProveedor =
          PROVEEDOR.CodProveedor
WHERE ARTICULO.CodLinea = @linea)
go

-- Invocando a la función
SELECT * FROM udf_ProveedorLineaArticulo(6)
go
```

Resultados | Mensajes

	CodProveedor	NomProveedor	Direccion	Ciudad
1	5	DISTRIBUIDORA ALBRICIAS	LEONCIO PRADO 625 MAGDALENA	LIMA
2	6	DISTRIBUIDORA DEL HOGAR	SAN MARTIN 1187 SMP	LIMA
3	11	QUIMICA DEL NORTE	JOSE CARLOS MARIATEGUI 473 CERCADO	CHICLAYO

4. FUNCIÓN TABLA EVALUADA MULTISENTENCIA

Es una función que ejecuta una ó más sentencias Transact-SQL, como en un procedimiento almacenado, y que retorna el resultado en formato de tabla. Una función de este tipo puede ser referenciada en la cláusula FROM de una sentencia SELECT, tal como se hace con una vista.

El tipo de valor de retorno debe ser **table**, y define el nombre y estructura de la tabla, siendo su ámbito solo local.

4.1. Creación de una función tabla evaluada multisentencia

Ejercicio 13.3: Tabla evaluada multisentencia

```
USE QhatuPERU
go

CREATE FUNCTION udf_ListaGuias(
     @salida varchar(15))
     RETURNS @lista table(
          Numero int NOT NULL PRIMARY KEY,
          Descripcion varchar(75) NOT NULL)
AS
BEGIN
     IF @salida = 'Tienda'
          INSERT @lista
          SELECT g.NumGuia, t.direccion
               FROM GUIA_ENVIO g
                    INNER JOIN TIENDA t
                    ON g.CodTienda = t.CodTienda
     ELSE IF @salida = 'Fecha'
          INSERT @lista
          SELECT NumGuia,
```

```
            CONVERT(CHAR(11), FechaSalida, 106)
            FROM GUIA_ENVIO
        RETURN
END
go
```

La función **udf_ListaGuias** retorna una variable de tipo **table** con dos columnas: **Numero** y **Descripcion**.

El contenido de la tabla está determinado por el argumento **@salida**. Si **@salida** contiene la cadena **'Tienda'**, llena la tabla con los números de guía y sus direcciones de envío; si **@salida** contiene la cadena **'Fecha'**, llena la tabla con los números de guía y sus respectivas fechas de emisión.

Para ejecutar la función:

```
SELECT * FROM dbo.udf_ListaGuias('Fecha')
go
```


Resultados Mensajes

	Numero	Descripcion
1	1	20 Mar 2013
2	2	25 Mar 2013
3	3	30 Mar 2013
4	4	09 Abr 2013
5	5	14 Abr 2013
6	6	20 Mar 2013
7	7	25 Mar 2013
8	8	30 Mar 2013
9	9	04 Abr 2013
10	10	09 Abr 2013

```
SELECT * FROM dbo.udf_ListaGuias('Tienda')
go
```

Resultados | Mensajes

	Numero	Descripcion
1	1	AV. LA PAZ 659
2	2	AV. BOLIVAR 1789
3	3	AV. SAENZ PEÑA 590
4	4	PANAMERICANA NORTE KM. 17.5
5	5	AV. ESPAÑA 775
6	6	AV. LA PAZ 659
7	7	AV. BOLIVAR 1789
8	8	AV. SAENZ PEÑA 590
9	9	PANAMERICANA NORTE KM. 17.5
10	10	AV. ESPAÑA 775

5. CREACIÓN DE UNA FUNCIÓN CON ENLACE DE ESQUEMA

Cuando una función se crea con la opción SCHEMABINDING, es enlazada a los objetos de base de datos a los que hace referencia. Estos objetos no podrán ser modificados (ALTER) ó eliminados (DROP).

Sintáxis

```
CREATE FUNCTION nombre_función(
   @parámetro1 tipo_dato [ = valor ] ,
   @parámetro2 tipo_dato [ = valor ] , ...
   RETURNS tipo_dato_valor_retorno
WITH SCHEMABINDING
[ AS ]
BEGIN
   sentencias_sql
   RETURN valor_retorno
END
```

Una función puede tener enlace de esquema solo si se cumplen las siguientes condiciones:

- Las funciones definidas por el usuario y las vistas referenciadas por la función tienen también enlace de esquema.
- Los objetos a los que la función hace referencia no son referenciados con la forma dueño.objeto.
- La función, y los objetos a los que hace referencia pertenecen a la misma base de datos.
- El usuario que crea la función tiene el permiso REFERENCE sobre todos los objetos a los que la función hace referencia.

6. EL OPERADOR APPLY

El operador APPLY permite invocar a una función tabla evaluada en línea por cada fila retornada por una consulta externa. La tabla evaluada en línea actúa como la consulta derecha de una subconsulta correlacionada, donde la consulta derecha es evaluada por cada fila de la consulta izquierda, y las filas obtenidas se combinan en el resultado final. El resultado producido por el operador APPLY es el conjunto de columnas en la consulta izquierda seguido del conjunto de columnas de la consulta derecha.

6.1. Formas de APPLY

- CROSS APPLY retorna solo las filas de la tabla izquierda para las que la función tabla evaluada en línea produce un conjunto de resultados.
- OUTER APPLY retorna las filas de la tabla izquierda para las que la función tabla evaluada en línea produce un conjunto de resultados, y además las filas de la tabla izquierda que no generan un conjunto de resultados.

Ejercicio 13.4: Preparación de las tablas a utilizar en el ejemplo de uso del operador APPLY

Ejecute el siguiente código en una ventana de consulta:

```
USE QhatuPERU
go

-- Creación de la tabla Colaborador
CREATE TABLE Colaborador(
     IdColaborador int PRIMARY KEY,
```

```
     Nombre varchar(40) not null,
     IdSuperior int null )
go

INSERT INTO Colaborador
     VALUES(1, 'Matsukawa Maeda, Sergio', NULL)
INSERT INTO Colaborador
     VALUES(2, 'Coronel Castillo, Gustavo', 1)
INSERT INTO Colaborador
     VALUES(3, 'Marcelo Villalobos, Ricardo', 1)
INSERT INTO Colaborador
     VALUES(4, 'Henostroza Macedo, Guino', 1)
INSERT INTO Colaborador
     VALUES(5, 'Lucho Lutgardo, Edgar', 2)
INSERT INTO Colaborador
     VALUES(6, 'Flores Manco, Julio', 2)
INSERT INTO Colaborador
     VALUES(7, 'Valencia Morales, Pedro', 3)
INSERT INTO Colaborador
     VALUES(8, 'Carrasco Muñoz, Joel', 3)
INSERT INTO Colaborador
     VALUES(9, 'Arista Arbildo, Nori', 3)
INSERT INTO Colaborador
     VALUES(10, 'Becerra Florez, Ursula', 4)
INSERT INTO Colaborador
     VALUES(11, 'Alcántara Cerna, Violeta', 7)
INSERT INTO Colaborador
     VALUES(12, 'Zegarra Zavaleta, Daniel', 7)
INSERT INTO Colaborador
     VALUES(13, 'Gonzales Villegas, Julio', 7)
INSERT INTO Colaborador
     VALUES(14, 'Guerra Guerra, Jorge', 11)
go

-- Creación de la tabla DEPARTAMENTO
CREATE TABLE Departamento(
     idDepartamento int PRIMARY KEY,
     nombre varchar(30) not null,
     idResponsable int null FOREIGN KEY
     REFERENCES Colaborador )
go

INSERT INTO Departamento VALUES(1,
     'Recursos Humanos', 2)
INSERT INTO Departamento VALUES(2,
```

```
     'Marketing', 7)
INSERT INTO Departamento VALUES(3,
     'Finanzas', 8)
INSERT INTO Departamento VALUES(4,
     'Investigación', 9)
INSERT INTO Departamento VALUES(5,
     'Capacitación', 4)
INSERT INTO Departamento VALUES(6,
     'Logística', NULL)
go
```

Consulte la tabla **Colaborador**:

```
SELECT * FROM Colaborador
go
```

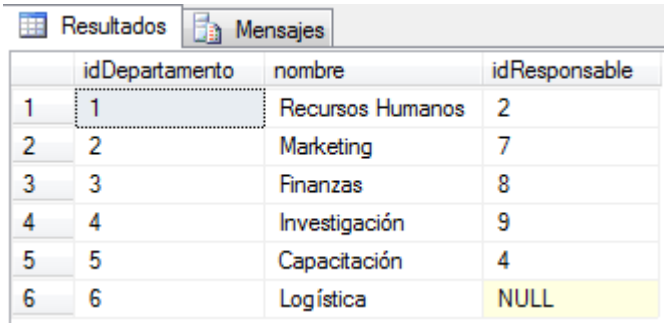
Resultados | Mensajes

	IdColaborador	Nombre	IdSuperior
1	1	Matsukawa Maeda, Sergio	NULL
2	2	Coronel Castillo, Gustavo	1
3	3	Marcelo Villalobos, Ricardo	1
4	4	Henostroza Macedo, Guino	1
5	5	Lucho Lutgardo, Edgar	2
6	6	Flores Manco, Julio	2
7	7	Valencia Morales, Pedro	3
8	8	Carrasco Muñoz, Joel	3
9	9	Arista Arbildo, Nori	3
10	10	Becerra Florez, Ursula	4

Note que la tabla contiene una lista de los colaboradores indicando para cada uno quién es su superior inmediato, el que a su vez también es un colaborador.

Ahora consulte la tabla **Departamento**:

```
SELECT * FROM Departamento
go
```



Resultados | Mensajes

	idDepartamento	nombre	idResponsable
1	1	Recursos Humanos	2
2	2	Marketing	7
3	3	Finanzas	8
4	4	Investigación	9
5	5	Capacitación	4
6	6	Logística	NULL

La tabla contiene una lista de los departamentos y muestra en la tercera columna el código del colaborador que es responsable del departamento.

Ejercicio 13.5: Uso del operador APPLY

Se desea obtener una lista que muestre para cada departamento a su colaborador responsable, y a los subordinados de cada colaborador responsable.

La función tabla evaluada en línea **udf_ListaSubordinados** utiliza una CTE (Common Table Expression) para construir una lista de los subordinados de un colaborador indicando en que nivel se encuentran respecto a dicho colaborador.

```
USE QhatuPERU
go

-- Función tabla evaluada en línea
-- que retorna todos los subordinados
-- del empleado que se le entrega como
-- argumento
CREATE FUNCTION udf_ListaSubordinados(
     @colaborador AS INT )
     RETURNS @lista TABLE(
          idColaborador int not null,
          nombre varchar(40)not null,
          idSuperior int null,
          nivel int not null )
AS
BEGIN
     WITH Arbol_Colaboradores(idColaborador,
               nombre, idSuperior, nivel)
```

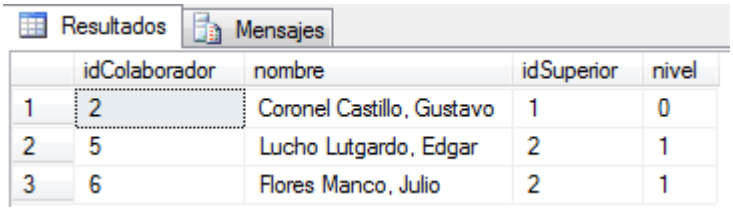
```
        AS
        (
        -- Miembro ancla
        SELECT idColaborador, nombre, idSuperior, 0
             FROM Colaborador
             WHERE idColaborador = @colaborador

        UNION all

        -- Miembro recursivo
        SELECT c.idColaborador, c.nombre,
             c.idSuperior, ac.nivel+1
        FROM Colaborador c INNER JOIN
             Arbol_Colaboradores ac
             ON c.idSuperior = ac.idColaborador
        )
        INSERT INTO @lista
        SELECT * FROM Arbol_Colaboradores
        RETURN
END
go
```

La siguiente consulta usa la función para obtener la lista de subordinados del colaborador de código **2**.

```
SELECT * FROM udf_ListaSubordinados(2)
go
```



Resultados | Mensajes

	idColaborador	nombre	idSuperior	nivel
1	2	Coronel Castillo, Gustavo	1	0
2	5	Lucho Lutgardo, Edgar	2	1
3	6	Flores Manco, Julio	2	1

Observe que la función muestra a todos los subordinados indicando para cada subordinado, cuántos niveles se encuentra por debajo del colaborador que se le entregó como argumento.

Para obtener la lista de responsables por departamento con sus subordinados, ejecute la siguente consulta APPLY que invoca a la función **udf_ListaSubordinados**.

```
-- Subordinados del responsable de cada departamento
SELECT * FROM Departamento d
     CROSS APPLY
     udf_ListaSubordinados(d.idResponsable) ls
go
```

Resultados | Mensajes

	idDepartamento	nombre	idResponsable	idColaborador	nombre	idSuperior	nivel
1	1	Recursos Humanos	2	2	Coronel Castillo, Gustavo	1	0
2	1	Recursos Humanos	2	5	Lucho Lutgardo, Edgar	2	1
3	1	Recursos Humanos	2	6	Flores Manco, Julio	2	1
4	2	Marketing	7	7	Valencia Morales, Pedro	3	0
5	2	Marketing	7	11	Alcántara Cema, Violeta	7	1
6	2	Marketing	7	12	Zegarra Zavaleta, Daniel	7	1
7	2	Marketing	7	13	Gonzales Villegas, Julio	7	1
8	2	Marketing	7	14	Guerra Guerra, Jorge	11	2
9	3	Finanzas	8	8	Carrasco Muñoz, Joel	3	0
10	4	Investigación	9	9	Arista Arbildo, Nori	3	0
11	5	Capacitación	4	4	Henostroza Macedo, Guino	1	0
12	5	Capacitación	4	10	Becerra Florez, Ursula	4	1

En el resultado no aparece el departamento **Logística** ya que éste no tiene registrado a su responsable. Para incluirlo en el resultado ejecute OUTER APPLY.

```
SELECT * FROM Departamento d
   OUTER APPLY
   udf_ListaSubordinados(d.idResponsable) ls
go
```

Resultados | Mensajes

	idDepartamento	nombre	idResponsable	idColaborador	nombre	idSuperior	nivel
1	1	Recursos Humanos	2	2	Coronel Castillo, Gustavo	1	0
2	1	Recursos Humanos	2	5	Lucho Lutgardo, Edgar	2	1
3	1	Recursos Humanos	2	6	Flores Manco, Julio	2	1
4	2	Marketing	7	7	Valencia Morales, Pedro	3	0
5	2	Marketing	7	11	Alcántara Cema, Violeta	7	1
6	2	Marketing	7	12	Zegarra Zavaleta, Daniel	7	1
7	2	Marketing	7	13	Gonzales Villegas, Julio	7	1
8	2	Marketing	7	14	Guerra Guerra, Jorge	11	2
9	3	Finanzas	8	8	Carrasco Muñoz, Joel	3	0
10	4	Investigación	9	9	Arista Arbildo, Nori	3	0
11	5	Capacitación	4	4	Henostroza Macedo, Guino	1	0
12	5	Capacitación	4	10	Becerra Florez, Ursula	4	1
13	6	Logística	NULL	NULL	NULL	NULL	NULL

7. EJERCICIOS PROPUESTOS

1. Escriba una función que entrega una lista de los artículos de la línea L, donde L es el nombre de la línea de artículos.

2. Escriba una función que entrega una lista de los artículos del proveedor P y de la línea L, donde P y L son el nombre del proveedor y el nombre de la línea respectivamente.

3. Crear una función que entregue el monto total de los artículos enviados a la tienda T, donde T es el código de la tienda.

4. Crear una función que entregue el monto total de los artículos enviados a la tienda T durante el mes M del año A.

5. Crear una función que entregue la lista de los N artículos más enviados a las tiendas; N es un entero.

6. Crear una función que entregue el balance entrada/salida del artículo A. El procedimiento debe entregar el total de unidades que ingresaron, y el total de unidades que salieron del artículo A.

7. Crear una función que entregue el monto total adquirido al proveedor P durante el año A, donde P es el código del proveedor.

8. Crear una función que calcule el monto promedio mensual enviado a la tienda T, donde T es el código de la tienda.