PROGRAMACIÓN DE BASE DE DATOS

Procedimientos Almacenados.

ING. JAIME LLANOS BARDALES

Introducción

- **Quiz**

Poll Everywhere

Agenda

Módulo 9: Funciones personalizadas

Objetivo: Manejo de funciones con el objetivo de crear funciones personalizadas.

- Uso de función Merge.
- Tipos de Funciones: Return Tables, Funciones Escalares
- Cross Apply y Outer Apply
- Concepto y tipos de procedimientos
- Creación, ejecución y eliminación de procedimientos.

Merge

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Permite con una misma sentencia realizar un UPDATE si el registro existe, o un INSERT si se trata de un nuevo registro.

- La sentencia puede paralelizarse de forma transparente.

- Se evita la necesidad de realizar actualizaciones múltiples.

- Es especialmente útil para realizar operaciones en masa y, como ya he mencionado, en aplicaciones de data warehousing.

```
MERGE <table_destino> [AS TARGET]
USING <table_origen> [AS SOURCE]
   ON <condicion_compara_llaves>
[WHEN MATCHED THEN
          <accion cuando coinciden> ]
[WHEN NOT MATCHED [BY TARGET] THEN
          <accion cuando no coinciden por destino> ]
[WHEN NOT MATCHED BY SOURCE THEN
          <accion cuando no coinciden por origen> ];
```

Merge ejemplos

En el presente ejemplo voy a partir del hecho de que tenemos dos tablas en un data warehouse, una de ellas con muchos registros denominada clientes y otra más pequeña denominada datos_cli cuyos registros deben insertarse en la tabla clientes.

```
MERGE INTO DBO.TMPJCD_CIENTE DEST
USING dbo.CL_CLIENTE ORIG
ON (ORIG.ID_CLIENTE = DEST.ID_CLIENTE)
WHEN MATCHED AND DEST.CLI_NOMBRES <> ORIG.CLI_NOMBRES   THEN
       UPDATE SET
       CLI_NOMBRES = ORIG.CLI_NOMBRES,
       CLI_APELLIDOS = ORIG.CLI_APELLIDOS,
       CLI_NOMBRE = ORIG.CLI_NOMBRE,
       CLI_DIRECCION = ORIG.CLI_DIRECCION,
       TELEFONOS = ORIG.TELEFONOS,
       FECHA_NAC = ORIG.FECHA_NAC,
       FECCARGA = GETDATE()
WHEN NOT MATCHED BY DEST THEN
       INSERT (ID_CLIENTE, CLI_NOMBRES, CLI_APELLIDOS,
       CLI_NOMBRE, CLI_DIRECCION, TELEFONOS, FECHA_NAC,
       FECCARGA)
       VALUES
       (ORIG.ID_CLIENTE, ORIG.CLI_NOMBRES, ORIG.CLI_APELLIDOS,
       ORIG.CLI_NOMBRE, ORIG.CLI_DIRECCION, ORIG.TELEFONOS, ORIG.FECHA_NAC,
       GETDATE())

       WHEN NOT MATCHED BY SOURCE THEN
              DELETE
       ;
```

Sentencias de control

IF THEN ELSE

El Bloque IF permite evaluar una condición que retorna **TRUE** para ejecutar una acción y **ELSE** para ejecutar en caso no cumpla la condición

```
IF
(SELECT COUNT(*) FROM [dbo].[countries]) = 0
PRINT 'No hay filas en la tabla Paises'
ELSE PRINT 'Hay filas en la tabla Paises' ;
```

```
1 %
Messages
Hay filas en la tabla Paises

Completion time: 2020-07-31T20:14:29.5967740-05:00
```

```
-- Uses AdventureWorks

DECLARE @maxWeight float, @productKey integer
SET @maxWeight = 100.00
SET @productKey = 424
IF @maxWeight <= (SELECT Weight from DimProduct WHERE ProductKey=@productKey)
    (SELECT @productKey, EnglishDescription, Weight, 'This product is too heav
ELSE
    (SELECT @productKey, EnglishDescription, Weight, 'This product is availabl
```

Sentencias de control

WHILE loop

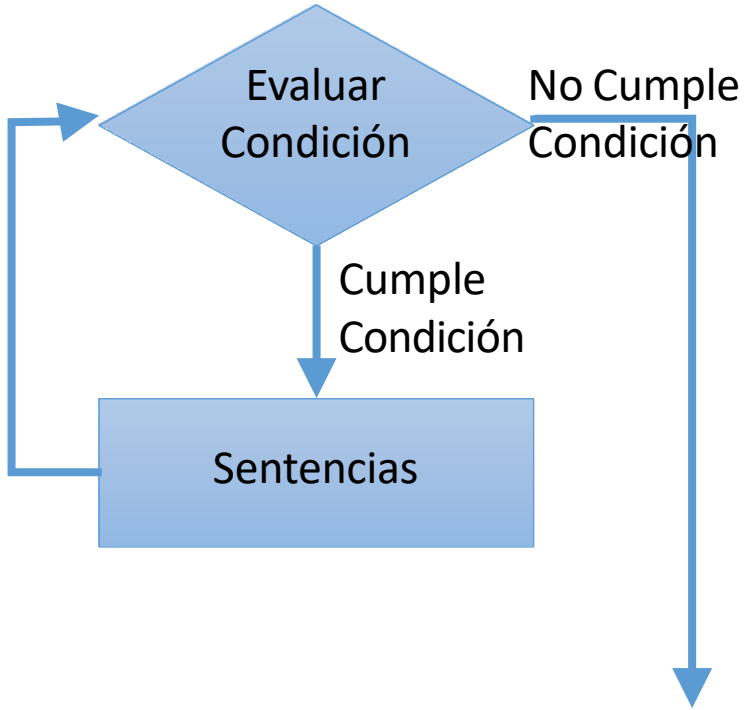
Establece una condición para la ejecución repetida de una instrucción SQL o bloque de instrucciones. Las declaraciones se ejecutan repetidamente siempre que la condición especificada sea verdadera.

```
DECLARE @Fila INT
SET @Fila=1
WHILE ( @Fila <= 10)
BEGIN
    PRINT '# Fila ' + CONVERT(VARCHAR,@Fila)
    SET @Fila  = @Fila  + 1
END
```

Messages

```
# Fila 1
# Fila 2
# Fila 3
# Fila 4
# Fila 5
# Fila 6
# Fila 7
# Fila 8
# Fila 9
# Fila 10
```

```
WHILE condition
BEGIN
    {...statements...}
END
```



Sentencias de control

Etiquetas

Altera el flujo de ejecución a una etiqueta. La instrucción o las instrucciones de Transact-SQL que siguen a GOTO se omiten y el procesamiento continúa en la etiqueta.

```
DECLARE @Fila int;
SET @Fila = 1;
WHILE @Fila < 10
BEGIN
    SELECT @Fila
    SET @Fila = @Fila + 1
    IF @Fila = 2 GOTO Salto_Uno -- Ejecuta el salto
    IF @Fila = 3 GOTO Salto_Dos -- No llega a este punto
END
Salto_Uno:
    SELECT '1- Salio del Bucle'
    GOTO Salto_Tres; -- Evita que se ejecute el Salto_Dos
Salto_Dos:
    SELECT '2- Salto Dos'
Salto_Tres:
    SELECT 'Salto Tres -- Fin';
```

Resultado

(No column name)
1

(No column name)
1- Salio del Bucle

(No column name)
Salto Tres -- Fin

Sentencias de control

Permiten el control de flujo de operaciones.

If Else

Nos Permite Ejecutar Instrucciones Condicionales.

While

Nos permite repetir la ejecución de un conjunto de instrucciones, mientras la condición sea verdadera

Case

La Sentencia Case Compara Un valor con una lista de valores y ejecuta una o más sentencias que corresponde. Y En Caso De No Cumplirse Devolverá Un Valor Por Defecto.

Sentencias de control

Permiten el control de flujo de operaciones.

IF <Expresion_Logica>
 <Instruccion>
ELSE
 <Instruccion>

If Else

```
IF (select count(1) from [Order Details] where
Quantity > 120) > 0
begin
print 'Pedidos con cantidades > 120'
select orderid, productid, quantity
from [Order Details]
where Quantity > 120
end
ELSE
print 'No se encontraron pedidos con cantidades
mayores a 120'
```

orderid	productid	quantity
10764	39	130
11072	64	130

Sentencias de control

Permiten el control de flujo de operaciones.

While <Expresion_Logica>
Begin
    <Instrucción Repetición>
End

While

```
Declare @FecIni Date, @FecFin Date
Declare @NroFilas Int
Set @FecIni = '1996-07-01'
Set @FecFin = '1996-07-07'
While @FecIni <= @FecFin
Begin
    select @NroFilas = count(1)
    from Orders where OrderDate = @FecIni
    PRINT convert(varchar(10), @FecIni)
    + ' Nro Pedidos ‘
    + convert(varchar(10),@NroFilas)
    set @FecIni = DateAdd(DAY,1,@FecIni)
End
```

```
1996-07-01 Nro Pedidos 0
1996-07-02 Nro Pedidos 0
1996-07-03 Nro Pedidos 0
1996-07-04 Nro Pedidos 1
1996-07-05 Nro Pedidos 1
1996-07-06 Nro Pedidos 0
1996-07-07 Nro Pedidos 0
```

Sentencias de control

Permiten el control de flujo de operaciones.

Case

```
select a.*,
    case when a.NroDias < 3 then 'En Tiempo'
    when a.NroDias < 6 then 'En Observacion'
    else 'En Revision'
    end FlgIndicador
from (
    select o.OrderID, o.OrderDate, o.ShippedDate,
    DATEDIFF (day, o.OrderDate, o.ShippedDate) NroDias
    from Orders o) a
```

Case
When <Condicion>
Then <Acciones x Verdad>
Else <Ninguna anterior>

	NroDias	FlgIndicador
0:00:00.000	12	En Revision
0:00:00.000	5	En Observacion
0:00:00.000	4	En Observacion
0:00:00.000	7	En Revision
0:00:00.000	2	En Tiempo
0:00:00.000	6	En Revision

Agenda

Módulo 10: Funciones personalizadas

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Objetivo: Manejo de funciones con el objetivo de crear funciones personalizadas.

- Tipos de Funciones: Return Tables, Funciones Escalares
- Cross Apply y Outer Apply
- Concepto y tipos de procedimientos
- Creación, ejecución y eliminación de procedimientos.

Variables

DECLARE

- Asignar un nombre. El nombre debe tener un único @ como primer carácter.
- Asignar un tipo de datos suministrado por el sistema o definido por el usuario y una longitud.
- Para las variables numéricas, se asignan también una precisión y una escala.
- Establecer el valor a NULL.

```
DECLARE @LastName nvarchar(30), @FirstName nvarchar(20), @StateProvince nchar(2);
```

Asignación desde valores

```
-- Declare two variables.
DECLARE @FirstNameVariable nvarchar(50),
   @PostalCodeVariable nvarchar(15);

-- Set their values.
SET @FirstNameVariable = N'Amy';
SET @PostalCodeVariable = N'BA5 3HX';
```

Asignación desde una instrucción SQL

```
DECLARE @EmpIDVariable int;

SELECT @EmpIDVariable = MAX(EmployeeID)
FROM HumanResources.Employee;
```

Bucles

Cuando deseamos repetir un conjunto de operaciones mientras no se cumpla una condición acude en nuestra ayuda el bucle

- Debe buscar una condición para lograr una ejecución repetida de una sentencia SQL o un bloque de sentencias.
- El bloque de sentencias se ejecutan repetidamente hasta que la condición especificada sea verdadera.
- Puede terminar el ciclo WHILE con las palabras clave BREAK y CONTINUE.

```
--Create 50000 random sales records
DECLARE @counter INT;
SET @counter = 1;
WHILE @counter <= 50000
    BEGIN
        INSERT INTO Sales
               SELECT NEWID(),
                      (ABS(CHECKSUM(NEWID())) % 4) + 1,
                      (ABS(CHECKSUM(NEWID())) % 2) + 1,
                      (ABS(CHECKSUM(NEWID())) % 9) + 1,
                     DATEADD(day, ABS(CHECKSUM(NEWID()) % 3650), '2020-04-01');
        SET @counter+=1;
    END;

SELECT * FROM [dbo].[vVentasPorVendedor];
SELECT * FROM dbo.[vVentasPorVendedor] WITH (NOEXPAND)
```

Bucles

```
DECLARE @Cont INT = 1;
WHILE(@Cont <= 10) BEGIN
select 'Hola Mundo';
set @Cont += 1;
END
```

Functions

Creación de funciones

```
IF
```

```
)UNT(*) FROM [dbo].[countries]) = 0
 hay filas en la tabla Paises'
[ 'Hay filas en la tabla Paises' ;
```

```
-- Transact-SQL Scalar Function Syntax
CREATE [ OR ALTER ] FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ][ type_schema_name. ] parameter_data_type
 [ = default ] [ READONLY ] }
    [ ,...n ]
  ]
)
RETURNS return_data_type
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    BEGIN
        function_body
        RETURN scalar_expression
    END
[ ; ]
```

```
-- Transact-SQL Inline Table-Valued Function Syntax
CREATE [ OR ALTER ] FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ] parameter_data_type
    [ = default ] [ READONLY ] }
    [ ,...n ]
  ]
)
RETURNS TABLE
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    RETURN [ ( ] select_stmt [ ) ]
[ ; ]
```

Functions

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Modificación de una función

- Modifica una función Transact-SQL o CLR existente, creada anteriormente por medio de la ejecución de la instrucción CREATE FUNCTION, sin cambiar los permisos y sin que afecte a ninguna otra función, procedimiento almacenado o desencadenador dependiente.

```
-- Transact-SQL Scalar Function Syntax
ALTER FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ][ type_schema_name. ] parameter_data_type
    [ = default ] }
    [ ,...n ]
  ]
)
RETURNS return_data_type
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    BEGIN
        function_body
        RETURN scalar_expression
    END
[ ; ]
```

Functions

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Eliminación de una función

- Quita una o más funciones definidas por el usuario de la base de datos actual. Las funciones definidas por el usuario se crean mediante CREATE FUNCTION y se modifican con ALTER FUNCTION.
- La función DROP admite funciones escalares definidas por el usuario

```
DROP FUNCTION [ IF EXISTS ] { [ schema_name. ] function_name } [ ,...n ]
[;]
```

DROP FUNCTION IF EXISTS sf_get_Cliente;

Functions

Ejecución de una función

```
USE [AdventureWorks2016CTP3]
GO

-- Declare a variable to return the results of the function.
DECLARE @ret nvarchar(15);

-- Execute the function while passing a value to the @status parameter
EXEC @ret = dbo.ufnGetSalesOrderStatusText
    @Status = 5;

-- View the returned value.  The Execute and Select statements must be executed at the sam
SELECT N'Order Status: ' + @ret;

-- Result:
-- Order Status: Shipped
```

Functions Example (Return Table)

```
CREATE FUNCTION [dbo].[FN_LISTA_VEND](@V_ANIO NUMERIC)
RETURNS TABLE
return
(
        SELECT
        FORMAT(P.FECPEDID,'yyyyMM') as PERIODO,
        PR.NOMCAT AS NOMCATEGORIA,
        PR.NOMSUBCAT AS NOMSUBCATEGORIA,
        PR.NOMPROD AS NOMPRODUCTO,
        SUM(P.MTOVALVTA) AS VENTAS,
        SUM(P.MTOBENEF) AS GANANCIAS,
        V.NOMVEND AS VENDEDOR,
        P.CODVEND AS CODVENDEDOR
        FROM
        HD_PEDIDO P
        LEFT JOIN MM_PRODUCTO PR ON P.CODPROD = PR.CODPROD
        LEFT JOIN MM_VENDEDOR V ON P.CODVEND = V.CODVEND
        WHERE FORMAT(P.FECPEDID,'yyyy')= @V_ANIO
        GROUP BY FORMAT(P.FECPEDID,'yyyyMM'),
        PR.NOMCAT,PR.NOMSUBCAT,
        PR.NOMPROD, V.NOMVEND,P.CODVEND
);
```

```
select * from FN_LISTA_VEND(2020)
```

	PERIODO	NOMCATEGORIA	NOMSUBCATEGORIA	NOMPRODUCTO	VENTAS	GANANCIAS	VENDEDOR	CODVENDEDOR
1	202001	Material de oficina	Almacenamiento	Adjustable Depth Letter/Legal Cart	1791.010	157.870	Autoventa	0
2	202001	Material de oficina	Almacenamiento	Adjustable Depth Letter/Legal Cart	3113.853	210.493	Christian Peña	2
3	202001	Material de oficina	Almacenamiento	Adjustable Personal File Tote	73.585	8.465	Manuel Gonzalez	3
4	202001	Material de oficina	Almacenamiento	Advantus Rolling Drawer Organizers	103.126	4.617	Christian Peña	2
5	202001	Material de oficina	Almacenamiento	Akro-Mils 12-Gallon Tote	98.009	8.639	Frank Llañes	1
6	202001	Material de oficina	Almacenamiento	Crate-A-Files	64.746	1.962	Manuel Gonzalez	3
7	202001	Material de oficina	Almacenamiento	Economy Rollaway Files	442.736	19.824	Christian Peña	2
8	202001	Material de oficina	Almacenamiento	Eldon Box, Blue	46.650	6.568	Autoventa	0
9	202001	Material de oficina	Almacenamiento	Eldon Box, Industrial	246.840	28.460	Autoventa	0
10	202001	Material de oficina	Almacenamiento	Eldon Box, Industrial	88.212	9.918	Manuel Gonzalez	3
11	202001	Material de oficina	Almacenamiento	Eldon Box, Single Width	9.449	2.360	Autoventa	0
12	202001	Material de oficina	Almacenamiento	Eldon File Cart, Industrial	939.500	86.900	Autoventa	0
13	202001	Material de oficina	Almacenamiento	Eldon File Cart, Industrial	939.941	7.623	Christian Peña	2
14	202001	Material de oficina	Almacenamiento	Eldon File Cart, Single Width	150.930	23.010	Autoventa	0
15	202001	Material de oficina	Almacenamiento	Eldon Folders, Industrial	281.400	2.040	Christian Peña	2
16	202001	Material de oficina	Almacenamiento	Eldon Folders, Industrial	303.800	12.800	Frank Llañes	1
17	202001	Material de oficina	Almacenamiento	Eldon Folders, Single Width	24.990	7.980	Christian Peña	2
18	202001	Material de oficina	Almacenamiento	Eldon Folders, Wire Frame	815.880	34.780	Autoventa	0
19	202001	Material de oficina	Almacenamiento	Eldon Gobal File Keepers	15.745	0.605	Autoventa	0

Functions Example (Return Table)

```
ALTER FUNCTION [dbo].[fn_top_vend_AUG](@V_PERIODO NUMERIC)
RETURNS TABLE
return (
WITH RUG  AS (
                SELECT   format(pe.fecpedid,'yyyyMM') as PERIODO,
                PR.NOMCAT,
                PR.NOMSUBCAT,
                PR.NOMPROD,
                sum(PE.MTOVALVTA) AS VENTAS,
                sum(PE.mtobenef) AS GANANCIAS,
                VE.CODVEND,
                VE.NOMVEND
                FROM HD_PEDIDO PE
                LEFT JOIN MM_PRODUCTO PR ON PE.CODPROD=PR.CODPROD
                LEFT JOIN MM_VENDEDOR VE ON PE.CODVEND=VE.CODVEND
                WHERE  FORMAT(PE.FECPEDID,'yyyyMM')= @V_PERIODO
                group by format(pe.fecpedid,'yyyyMM') ,
                        PR.NOMCAT, PR.NOMSUBCAT, PR.NOMPROD,
                PR.NOMSUBCAT, PR.NOMPROD, VE.CODVEND,VE.NOMVEND
                )
select TOP 3 a.NOMVEND, a.NOMCAT, a.NOMPROD, a.VENTAS, a.GANANCIAS
from RUG A
where a.CODVEND>0
order by A.VENTAS DESC)
```

```
select * from fn_top_vend_AUG(202201)
```

	NOMVEND	NOMCAT	NOMPROD	VENTAS	GANANCIAS
1	Manuel Gonzalez	Mobiliario	Chromcraft Bull-Nose Wood Oval Conference Tables...	63318.620	-3724.624
2	Victor Medina	Material de oficina	Hamilton Beach Stove, White	43941.000	162.120
3	Francisco Gutierrez	Mobiliario	Harbour Creations Executive Leather Armchair, Black	39072.660	868.120

Functions Example (Scalar Functions)

```
ALTER FUNCTION [dbo].[FN_DIF_FECHA] (@fecha date) returns varchar(200)
as
BEGIN
DECLARE @ANIOS AS NUMERIC,
@MESES AS NUMERIC, @DIAS AS NUMERIC, @RESULTADO AS VARCHAR(200)

SET @ANIOS = DATEDIFF(MONTH, @fecha, GETDATE()) / 12
SET @MESES = DATEDIFF(MONTH, @fecha, GETDATE()) % 12
SET @DIAS = (DATEDIFF(DAY, @fecha , GETDATE()) % 365) % 30
SET @RESULTADO = CAST(@ANIOS AS VARCHAR(3)) + ' Año(s) '+ CAST(@MESES
AS VARCHAR) + ' Mes(es) ' + CAST(@DIAS AS VARCHAR) + ' Día(s)'

RETURN @RESULTADO
END;
```

```
SELECT
E.ID_EMPLEADO,
E.NOMBRES,
E.APELLIDOS,
E.SUELDO,
ISNULL(E.PCT_COMISION,0) AS NULLTreat,
E.FECHA_INGRESO,
DBO.FN_DIF_FECHA(E.FECHA_INGRESO) AS DIFFECHA
FROM CL_EMPLEADOS E;
```

Results | Messages

	ID_EMPLEADO	NOMBRES	APELLIDOS	SUELDO	NULLTreat	FECHA_INGRESO	DIFFECHA
1	100	STEVEN	KING	24000.00	0.00	2021-07-05 00:00:00.000	1 Año(s) 9 Mes(es) 10 Día(s)
2	101	NEENA	KOCHHAR	17000.00	0.00	2014-10-05 00:00:00.000	8 Año(s) 6 Mes(es) 10 Día(s)
3	102	LEX	DE HAAN	17000.00	0.00	2011-03-23 00:00:00.000	12 Año(s) 1 Mes(es) 22 Día(s)
4	103	ALEXANDER	HUNOLD	9000.00	0.00	2016-11-19 00:00:00.000	6 Año(s) 5 Mes(es) 24 Día(s)
5	104	BRUCE	ERNST	6000.00	0.00	2012-04-26 00:00:00.000	11 Año(s) 0 Mes(es) 22 Día(s)
6	105	DAVID	AUSTIN	4800.00	0.00	2015-05-27 00:00:00.000	7 Año(s) 11 Mes(es) 21 Día(s)
7	106	VALLI	PATABALLA	4800.00	0.00	2013-10-27 00:00:00.000	9 Año(s) 6 Mes(es) 18 Día(s)
8	107	DIANA	LORENTZ	4200.00	0.00	2010-06-10 00:00:00.000	12 Año(s) 10 Mes(es) 8 Día(s)
9	108	NANCY	GREENBERG	12000.00	0.00	2020-06-02 00:00:00.000	2 Año(s) 10 Mes(es) 13 Día(s)
10	109	DANIEL	FAVIET	9000.00	0.00	2009-09-19 00:00:00.000	13 Año(s) 7 Mes(es) 27 Día(s)
11	110	JOHN	CHEN	8200.00	0.00	2016-08-04 00:00:00.000	6 Año(s) 8 Mes(es) 11 Día(s)
12	111	JUAN	SCIARRA	7700.00	0.00	2022-01-02 00:00:00.000	1 Año(s) 3 Mes(es) 9 Día(s)
13	112	JULIO	URMAN	7800.00	0.00	2010-06-06 00:00:00.000	12 Año(s) 10 Mes(es) 12 Día(s)
14	113	LUIS	POPP	6900.00	0.00	2020-05-16 00:00:00.000	2 Año(s) 11 Mes(es) 0 Día(s)
15	114	DEN	RAPHAELY	11000.00	0.00	2010-02-21 00:00:00.000	13 Año(s) 2 Mes(es) 22 Día(s)
16	115	ALEXANDER	KHOO	3100.00	0.00	2016-02-12 00:00:00.000	7 Año(s) 2 Mes(es) 0 Día(s)
17	116	SHELLI	BAIDA	2900.00	0.00	2021-09-10 00:00:00.000	1 Año(s) 7 Mes(es) 3 Día(s)
18	117	SIGAL	TOBIAS	2800.00	0.00	2013-11-21 00:00:00.000	9 Año(s) 5 Mes(es) 23 Día(s)
19	118	GUY	HIMURO	2600.00	0.00	2021-12-31 00:00:00.000	1 Año(s) 4 Mes(es) 11 Día(s)

Functions Example (Scalar Functions)

```
ALTER  FUNCTION [dbo].[FN_RANGOETAREO] (@fecha date) returns
varchar(40) as
BEGIN
DECLARE @EDAD AS NUMERIC,  @RESULTADO AS VARCHAR(200)

SET @EDAD = DATEDIFF(MONTH, @fecha, GETDATE()) / 12

IF @EDAD>= 18 AND @EDAD<=22
 BEGIN
 SET @RESULTADO = 'Generación Z'
 END
 ELSE IF @EDAD>=23 AND @EDAD<=35
  BEGIN
  SET @RESULTADO = 'Millenials'
  END
 ELSE IF @EDAD>=36 AND @EDAD<=53
  BEGIN
    SET @RESULTADO = 'Generación X'
  END
 ELSE IF @EDAD>=54 AND @EDAD<=71
  BEGIN
    SET @RESULTADO = 'Baby Bommers'
  END
  ELSE
BEGIN
SET @RESULTADO = 'NA'
END

RETURN @RESULTADO
END;
```

```
SELECT
CL.ID_CLIENTE,
CL.CLI_APELLIDOS,
CL.CLI_NOMBRE,
CL.FECHA_NAC,
DBO.FN_RANGOETAREO(CL.FECHA_NAC) AS
RANGOETAREO
FROM CL_CLIENTE CL;
```

Results | Messages

	ID_CLIENTE	CLI_APELLIDOS	CLI_NOMBRE	FECHA_NAC	RANGOETAREO
1	1667206	RIDGEWAY	RAMSAY RIDGEWAY	1983-06-18 00:00:00.000	Generación X
2	1667208	PEREZ	JUAN PEREZ	1985-08-24 00:00:00.000	Generación X
3	1894456	LINDEGREEN	MARCUS LINDEGREEN	1979-06-17 00:00:00.000	Generación X
4	1894457	GERALT	MADELENA GERALT	1983-01-01 00:00:00.000	Generación X
5	1900912	LEASE	BLANCHE LEASE	1981-07-20 00:00:00.000	Generación X
6	1906483	DUNHILL	BENEDICT DUNHILL	1982-01-02 00:00:00.000	Generación X
7	1923523	MALONE	LIANE MALONE	1984-03-10 00:00:00.000	Generación X
8	1927318	JEFFREYS	LEONA JEFFREYS	1987-08-22 00:00:00.000	Millenials
9	1929292	BALDWIN	DENYS BALDWIN	1983-01-02 00:00:00.000	Generación X
10	1930653	GILBOY	LUCAS GILBOY	1984-07-20 00:00:00.000	Generación X
11	1930743	BARNES	ROSETTA BARNES	1982-02-06 00:00:00.000	Generación X
12	1930752	KRIDER	POLLYANNA KRIDER	1983-03-08 00:00:00.000	Generación X
13	1939873	KIDWELL	MILBURN KIDWELL	1980-06-18 00:00:00.000	Generación X
14	1947434	OWENS	ROLPH OWENS	1979-01-01 00:00:00.000	Generación X
15	2026248	DAHL	ROSWALD DAHL	1986-05-14 00:00:00.000	Generación X
16	2031426	FILBERT	MYRON FILBERT	1982-09-26 00:00:00.000	Generación X
17	2032088	ZIMMERMAN	LYLYBEL ZIMMERMAN	1987-07-20 00:00:00.000	Millenials
18	2038830	BERRY	IDETTE BERRY	1986-01-04 00:00:00.000	Generación X
19	2060974	ROBEY	JUAN ROBEY	1985-04-11 00:00:00.000	Generación X

CROSS APPLY and OUTER APPLY in SQL Server

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

SQL Server admite funciones con valores de tabla, que son funciones que devuelven datos en forma de tablas.

Las operaciones JOIN en SQL Server se utilizan para unir dos o más tablas. Sin embargo, las operaciones JOIN no se pueden usar para unir una tabla con la salida de una función con valores de tabla.

Vimos cómo los operadores JOIN unen los resultados de dos tablas. Sin embargo, como se mencionó anteriormente, no se pueden usar para unir una función con valores de tabla con una tabla. Una función con valores de tabla es una función que devuelve registros en forma de tabla.

Primero escribamos una función simple con valores de tabla que acepte la identificación del autor como parámetro y devuelva todos las coincidencias.

CROSS APPLY Example in SQL Server

```
SELECT
E.APELLIDOS,
E.NOMBRES,
E.SUELDO,
E.ID_DPTO,
D.NOMBRE_DPTO
FROM
CL_EMPLEADOS E
INNER JOIN CL_DEPARTAMENTOS D ON E.ID_DPTO = D.ID_DPTO
```

Results | Messages

	APELLIDOS	NOMBRES	SUELDO	ID_DPTO	NOMBRE_DPTO
1	KING	STEVEN	24000.00	90	EXECUTIVE
2	KOCHHAR	NEENA	17000.00	90	EXECUTIVE
3	DE HAAN	LEX	17000.00	90	EXECUTIVE
4	HUNOLD	ALEXANDER	9000.00	60	IT
5	ERNST	BRUCE	6000.00	60	IT
6	AUSTIN	DAVID	4800.00	60	IT
7	PATABALLA	VALLI	4800.00	60	IT
8	LORENTZ	DIANA	4200.00	60	IT
9	GREENBERG	NANCY	12000.00	100	FINANCE
10	FAVIET	DANIEL	9000.00	100	FINANCE
11	CHEN	JOHN	8200.00	100	FINANCE
12	SCIARRA	JUAN	7700.00	100	FINANCE
13	URMAN	JULIO	7800.00	100	FINANCE

121 %

Results | Messages

	APELLIDOS	NOMBRES	SUELDO	ID_DPTO	NOMBRE_DPTO
1	KING	STEVEN	24000.00	90	EXECUTIVE
2	KOCHHAR	NEENA	17000.00	90	EXECUTIVE
3	DE HAAN	LEX	17000.00	90	EXECUTIVE
4	HUNOLD	ALEXANDER	9000.00	60	IT
5	ERNST	BRUCE	6000.00	60	IT
6	AUSTIN	DAVID	4800.00	60	IT
7	PATABALLA	VALLI	4800.00	60	IT
8	LORENTZ	DIANA	4200.00	60	IT
9	GREENBERG	NANCY	12000.00	100	FINANCE
10	FAVIET	DANIEL	9000.00	100	FINANCE
11	CHEN	JOHN	8200.00	100	FINANCE
12	SCIARRA	JUAN	7700.00	100	FINANCE
13	URMAN	JULIO	7800.00	100	FINANCE

```
ALTER FUNCTION fnobtenerdpto_jcd (@dpto int)

RETURNS TABLE

AS

RETURN

(

SELECT D.ID_DPTO, D.NOMBRE_DPTO, D.ID_LOCALIDAD FROM
CL_DEPARTAMENTOS D

WHERE D.ID_DPTO = @dpto

)

SELECT

E.APELLIDOS, E.NOMBRES, E.SUELDO, E.ID_DPTO,

D.NOMBRE_DPTO

FROM CL_EMPLEADOS E CROSS APPLY
fnobtenerdpto_jcd(E.ID_DPTO) D
```

OUTER APPLY Example in SQL Server


```
SELECT
E.APELLIDOS,
E.NOMBRES,
E.SUELDO,
E.ID_DPTO,
D.NOMBRE_DPTO
FROM
CL_EMPLEADOS E
LEFT JOIN CL_DEPARTAMENTOS D ON E.ID_DPTO = D.ID_DPTO
```

```
SELECT
E.APELLIDOS, E.NOMBRES, E.SUELDO, E.ID_DPTO,
D.NOMBRE_DPTO
FROM CL_EMPLEADOS E OUTER APPLY
fnobtenerdpto_jcd(E.ID_DPTO) D
```

Results | Messages

	APELLIDOS	NOMBRES	SUELDO	ID_DPTO	NOMBRE_DPTO
1	KING	STEVEN	24000.00	90	EXECUTIVE
2	KOCHHAR	NEENA	17000.00	90	EXECUTIVE
3	DE HAAN	LEX	17000.00	90	EXECUTIVE
4	HUNOLD	ALEXANDER	9000.00	60	IT
5	ERNST	BRUCE	6000.00	60	IT
6	AUSTIN	DAVID	4800.00	60	IT
7	PATABALLA	VALLI	4800.00	60	IT
8	LORENTZ	DIANA	4200.00	60	IT
9	GREENBERG	NANCY	12000.00	100	FINANCE
10	FAVIET	DANIEL	9000.00	100	FINANCE
11	CHEN	JOHN	8200.00	100	FINANCE
12	SCIARRA	JUAN	7700.00	100	FINANCE
13	URMAN	JULIO	7800.00	100	FINANCE

121 % | Results | Messages

	APELLIDOS	NOMBRES	SUELDO	ID_DPTO	NOMBRE_DPTO
1	KING	STEVEN	24000.00	90	EXECUTIVE
2	KOCHHAR	NEENA	17000.00	90	EXECUTIVE
3	DE HAAN	LEX	17000.00	90	EXECUTIVE
4	HUNOLD	ALEXANDER	9000.00	60	IT
5	ERNST	BRUCE	6000.00	60	IT
6	AUSTIN	DAVID	4800.00	60	IT
7	PATABALLA	VALLI	4800.00	60	IT
8	LORENTZ	DIANA	4200.00	60	IT
9	GREENBERG	NANCY	12000.00	100	FINANCE
10	FAVIET	DANIEL	9000.00	100	FINANCE
11	CHEN	JOHN	8200.00	100	FINANCE
12	SCIARRA	JUAN	7700.00	100	FINANCE
13	URMAN	JULIO	7800.00	100	FINANCE

Store Procedures

- Los procedimientos almacenados en SQL son un conjunto de instrucciones de tipo Transact-SQL o un método de acceso a los datos
- Dichas instrucciones se almacenan de forma física con un nombre dentro de la base de datos.

Como unidad de programación en otros lenguajes tienen capacidades:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida
- Contienen instrucciones de programación que realicen operaciones en la base de datos.
- Pueden contener llamadas a otros procedimientos.
- Devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos.

Store Procedures

Tipos de procedimientos almacenados

Tipos Procedimientos	Descripción
Definidos por el usuario	Definido por el usuario, se almacenan en una o mas base de datos
Temporales	Definidos por el usuario, solo que residen en la BD TempDB. Hay dos tipos de procedimientos temporales: locales y globales. Se diferencian entre sí por los nombres, la visibilidad y la disponibilidad. Los procedimientos temporales locales tienen como primer carácter de sus nombres un solo signo de número (#); solo son visibles en la conexión actual del usuario y se eliminan cuando se cierra la conexión. Los procedimientos temporales globales presentan dos signos de número (##) antes del nombre; son visibles para cualquier usuario después de su creación y se eliminan al final de la última sesión en la que se usa el procedimiento.
Sistema	Los procedimientos del sistema se incluyen con SQL Server. Están almacenados físicamente en la base de datos interna y oculta *Resource* y se muestran de forma lógica en el esquema **sys**

Store Procedures

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Creación de un Store Procedure

- Se requiere contar con los permisos para crear estos objetos
- Se puede especificar la información relevante a la construcción del Store
- Se puede especificar los parámetros de entrada o salida
- Se debe especificar una operación o validación en el cuerpo del store

```
-- ============================================
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- ============================================
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
    -- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

Store Procedures

Eliminación de un Store Procedure

- Eliminar un procedimiento puede hacer que los objetos y scripts dependientes produzcan un error
- Se requiere contar con el permiso ALTER en el esquema al que pertenece el procedimiento o el permiso CONTROL en el procedimiento.

```
DROP PROCEDURE <stored procedure name>;
GO
```

Store Procedures

Ejecución de un Store Procedure

- Hay dos formas diferentes de ejecutar un procedimiento almacenado. El primer método y más común es que una aplicación o un usuario llame al procedimiento. El segundo método consiste en establecer el procedimiento para que se ejecute automáticamente cuando se inicie una instancia de SQL Server

```
USE AdventureWorks2012;
GO
EXEC dbo.uspGetEmployeeManagers @BusinessEntityID = 50;
```

Store Procedures Ejemplos

```
ALTER PROCEDURE [dbo].[SP_CARGA_MES]
@PERIODO NUMERIC(6) --- VARIABLE DE TIPO NUMERICO DE 6 ENTEROS 202201
AS
BEGIN TRY  -- INICIA LA LOGICA
IF (SELECT COUNT(*) FROM HD_PEDIDO P WHERE FORMAT(P.FECPEDID,'yyyyMM') = @PERIODO) = 0
-- CUENTO SI EXISTEN REGISTROS PARA EL PERIODO QUE COINCIDA CON LA VARIABLE
        RAISERROR ('NO HAY REGISTROS', 11, 1)  -- INSTRUCCION PARA FORZAR UN ERROR

insert into RESVTAMES_JCD (PERIODO, PAIS, ESTADO, CIUDAD, VENTAS, GANANCIAS,
FECACTUALIZ)  -- CUERPO DEL PROGRAMA
SELECT
FORMAT(P.FECPEDID,'yyyyMM') as PERIODO,  -- POBLAR LA TABLA RESVTAMES_JCD CON LA DATA
DEL PERIDO QUE LE INDIQUEMOS
CL.NOMPAIS, CL.NOMESTADO, CL.NOMCIUDAD,
SUM(P.MTOVALVTA) AS VENTAS, -- FUNCIONES AGREGADAS
SUM(P.MTOBENEF) AS GANANCIAS,-- FUNCION AGREGADAS, GETDATE()
FROM HD_PEDIDO P
LEFT JOIN MM_CLIENTE CL ON P.CODCLI = CL.CODCLI
WHERE FORMAT(P.FECPEDID,'yyyyMM') = @PERIODO
GROUP BY FORMAT(P.FECPEDID,'yyyyMM'), CL.NOMPAIS,CL.NOMESTADO, CL.NOMCIUDAD;
END TRY

BEGIN CATCH -- SECCION PARA CONTROLAR ERRORES
    INSERT INTO dbo.AUDITORIA
    VALUES
  (SUSER_SNAME(), ERROR_NUMBER(),
   ERROR_STATE(), ERROR_SEVERITY(),
   ERROR_LINE(), ERROR_PROCEDURE(),
   ERROR_MESSAGE(),  GETDATE());

   DECLARE @Message varchar(MAX) = ERROR_MESSAGE(),
        @Severity int = ERROR_SEVERITY(),
        @State smallint = ERROR_STATE()

   RAISERROR (@Message, @Severity, @State)
  END CATCH;
```

```
USE [taller_SQL]
GO
DECLARE@return_value int
EXEC@return_value = [dbo].[SP_CARGA_MES]
@PERIODO = 202202
SELECT'Return Value' = @return_value

GO

SELECT * FROM AUDITORIA;
```

121 %

Results | Messages

	PERIODO	PAIS	ESTADO	CIUDAD	VENTAS	GANANCIAS	FECACTUALIZ
1	202202	Afganistán	Kabul	Kabul	8755.770	497.640	2023-04-11 15:45:55.143
2	202202	Alemania	Baden-Wurtemberg	Stuttgart	169.500	8.250	2023-04-11 15:45:55.143
3	202202	Alemania	Baden-Wurtemberg	Ulm	5345.700	151.830	2023-04-11 15:45:55.143
4	202202	Alemania	Baja Sajonia	Oldenburg	4540.440	171.480	2023-04-11 15:45:55.143
5	202202	Alemania	Bavaria	Aschaffenburg	15436.269	481.317	2023-04-11 15:45:55.143
6	202202	Alemania	Bavaria	Erlangen	855.960	91.680	2023-04-11 15:45:55.143
7	202202	Alemania	Bavaria	Munich	26433.840	1425.960	2023-04-11 15:45:55.143
8	202202	Alemania	Bavaria	Nuremberg	984.389	29.934	2023-04-11 15:45:55.143
9	202202	Alemania	Bavaria	Rosenheim	2495.040	165.120	2023-04-11 15:45:55.143
10	202202	Alemania	Berlín	Berlín	5963.715	440.280	2023-04-11 15:45:55.143
11	202202	Alemania	Bremen	Bremen	699.764	102.048	2023-04-11 15:45:55.143
12	202202	Alemania	Hamburgo	Hamburgo	56644.890	3356.730	2023-04-11 15:45:55.143
13	202202	Alemania	Hesse	Offenbach	351.690	40.440	2023-04-11 15:45:55.143
14	202202	Alemania	Renania del Norte-Westfalia	Bielefeld	1066.320	50.640	2023-04-11 15:45:55.143
15	202202	Alemania	Renania del Norte-Westfalia	Bochum	442.500	13.500	2023-04-11 15:45:55.143
16	202202	Alemania	Renania del Norte-Westfalia	Cologne	182.400	12.720	2023-04-11 15:45:55.143
17	202202	Alemania	Renania del Norte-Westfalia	Dormagen	14923.350	860.790	2023-04-11 15:45:55.143
18	202202	Alemania	Renania del Norte-Westfalia	Dortmund	2197.500	170.280	2023-04-11 15:45:55.143
19	202202	Alemania	Renania del Norte-Westfalia	Duisburg	5033.820	121.860	2023-04-11 15:45:55.143

Referencias

Create Function
https://docs.microsoft.com/es-es/sql/t-sql/statements/create-function-transact-sql?view=sql-server-ver15

Alter Function
https://docs.microsoft.com/es-es/sql/t-sql/statements/alter-function-transact-sql?view=sql-server-ver15

Drop Function
https://docs.microsoft.com/es-es/sql/t-sql/statements/drop-function-transact-sql?view=sql-server-ver15

Exec Function
https://docs.microsoft.com/es-es/sql/relational-databases/user-defined-functions/execute-user-defined-functions?view=sql-server-ver15

Herramientas de Trabajo

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Online

SQL Online

SQL Server Management Studio

v. 19.1

Microsoft SQL Server

Fin de la sesión