FUNCTIONS

Sagar Giri

SQL User-Defined Functions

Functions in programming languages are subroutines used to encapsulate frequently performed logic.
Any code that must perform the logic incorporated in a function can call the function rather than having to repeat all of the function logic.

Types:

- Built-in functions
- **User-defined functions**
  - User-defined functions take zero or more input parameters, and return a single value. Some user-defined functions return a single, scalar data value, such as an int, char, or decimal value.

Creating a Function

A standalone function is created using the

CREAT E FUNCT ION statement.

```
CREATE FUNCTION CubicVolume
-- Input dimensions in centimeters.
   (@CubeLength decimal(4,1), @CubeWidth decimal(4,1),
    @CubeHeight decimal(4,1) )
RETURNS decimal(12,3) -- Cubic Centimeters.
AS
BEGIN
   RETURN ( @CubeLength * @CubeWidth * @CubeHeight )
END
```

```
CREATE TABLE Bricks
    (
     BrickPartNmbr    int PRIMARY KEY,
     BrickColor       nchar(20),
     BrickHeight      decimal(4,1),
     BrickLength      decimal(4,1),
     BrickWidth       decimal(4,1),
     BrickVolume AS
                (
                 dbo.CubicVolume(BrickHeight,
                                 BrickLength, BrickWidth)
                )
    )
```

- **SQL Server 2000 also supports user-defined functions that return a table data type:**
- A function can declare an internal table variable, insert rows into the variable, and then return the variable as its return value.
- A class of user-defined functions known as in-line functions, return the result set of a SELECT statement as a variable of type table.

```
CREATE FUNCTION fn_getdeliveryprice
   (p_country Varchar(50), p_city Varchar(50)) RETURNS Float
BEGIN
DECLARE v_price Float;
/*
for not identified customer's location delivery price is 45 euro
*/
SET v_price = 45;

IF p_country = 'iceland' THEN
   SET v_price=8.15;
END IF;
IF p_country = 'poland'
  IF p_city = 'warsaw'
   SET v_price=7.85;
  ELSE
   SET v_price=9.75;
  END IF;
END IF;

/* ... here more price options can be placed ... */

RETURN v_price;

END
```

Then we can call this function using single SELECT:

```
SELECT fn_getdeliveryprice('poland', 'warsaw');
```

Or call function for every row in SELECT statement:

```
SELECT CustomerID, CustomerName,
fn_getdeliveryprice(Country, City) AS DelivPay
FROM Customers
WHERE CustomerID BETWEEN 10 AND 12;
```

The results will came as following:

CustomerID	CustomerName	DelivPay
10	Erika Nass	8.14
11	Somerset Dogan	45
12	Famke Bacher	45

Benefits of UserDefined Functions?

- we can use these functions in so many different places when compared to the SQL Server stored procedure.
- The ability for a function to act like a table (for Inline table and Multi-statement table functions) gives developers the ability to break out complex logic into shorter and shorter code blocks.
- Gives additional benefit of making the code less complex and easier to write and maintain.
- In the case of a Scalar User-Defined Function, the ability to use this function anywhere you can use a scalar of the same data type is also a very powerful thing.

Conclusion:

- Combining these advantages with the ability to pass parameters into these database objects makes the SQL Server User-Defined function a very powerful tool.

THANK YOU!