Chapter 43: Functions (Scalar/Single Row)

SQL provides several built-in scalar functions. Each scalar function takes one value as input and returns one value as output for each row in a result set.

You use scalar functions wherever an expression is allowed within a T-SQL statement.

Section 43.1: Date And Time

In SQL, you use date and time data types to store calendar information. These data types include the time, date, smalldatetime, datetime, datetime2, and datetimeoffset. Each data type has a specific format.

Data type	Format
time	hh:mm:ss[.nnnnnnn]
date	YYYY-MM-DD
smalldatetime	YYYY-MM-DD hh:mm:ss
datetime	YYYY-MM-DD hh:mm:ss[.nnn]
datetime2	YYYY-MM-DD hh:mm:ss[.nnnnnnn]
datetimeoffset	YYYY-MM-DD hh:mm:ss[.nnnnnnn] [+/-]hh:mm

The `DATENAME` function returns the name or value of a specific part of the date.

```
SELECT DATENAME (weekday,'2017-01-14') as Datename
```

Datename
Saturday

You use the `GETDATE` function to determine the current date and time of the computer running the current SQL instance. This function doesn't include the time zone difference.

```
SELECT GETDATE() as Systemdate
```

Systemdate
2017-01-14 11:11:47.7230728

The `DATEDIFF` function returns the difference between two dates.

In the syntax, datepart is the parameter that specifies which part of the date you want to use to calculate difference. The datepart can be year, month, week, day, hour, minute, second, or millisecond. You then specify the start date in the startdate parameter and the end date in the enddate parameter for which you want to find the difference.

```
SELECT SalesOrderID, DATEDIFF(day, OrderDate, ShipDate)
AS 'Processing time'
FROM Sales.SalesOrderHeader
```

SalesOrderID	Processing time
43659	7
43660	7
43661	7
43662	7

The `DATEADD` function enables you to add an interval to part of a specific date.

```
SELECT DATEADD (day, 20, '2017-01-14') AS Added20MoreDays
```

Added20MoreDays
2017-02-03 00:00:00.000

Section 43.2: Character modifications

Character modifying functions include converting characters to upper or lower case characters, converting numbers to formatted numbers, performing character manipulation, etc.

The `lower(char)` function converts the given character parameter to be lower-cased characters.

```
SELECT customer_id, lower(customer_last_name) FROM customer;
```

would return the customer's last name changed from "SMITH" to "smith".

Section 43.3: Configuration and Conversion Function

An example of a configuration function in SQL is the `@@SERVERNAME` function. This function provides the name of the local server that's running SQL.

```
SELECT @@SERVERNAME AS 'Server'
```

Server
SQL064

In SQL, most data conversions occur implicitly, without any user intervention.

To perform any conversions that can't be completed implicitly, you can use the `CAST` or `CONVERT` functions.

The `CAST` function syntax is simpler than the `CONVERT` function syntax, but is limited in what it can do.

In here, we use both the `CAST` and `CONVERT` functions to convert the datetime data type to the `varchar` data type.

The `CAST` function always uses the default style setting. For example, it will represent dates and times using the format YYYY-MM-DD.

The `CONVERT` function uses the date and time style you specify. In this case, 3 specifies the date format dd/mm/yy.

```
USE AdventureWorks2012
GO
SELECT FirstName + ' ' + LastName + ' was hired on ' +
    CAST(HireDate AS varchar(20)) AS 'Cast',
    FirstName + ' ' + LastName + ' was hired on ' +
    CONVERT(varchar, HireDate, 3) AS 'Convert'
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
ON p.BusinessEntityID = e.BusinessEntityID
GO
```

Cast	Convert
David Hamiltion was hired on 2003-02-04	David Hamiltion was hired on 04/02/03

Another example of a conversion function is the `PARSE` function. This function converts a string to a specified data type.

In the syntax for the function, you specify the string that must be converted, the `AS` keyword, and then the required

data type. Optionally, you can also specify the culture in which the string value should be formatted. If you don't specify this, the language for the session is used.

If the string value can't be converted to a numeric, date, or time format, it will result in an error. You'll then need to use `CAST` or `CONVERT` for the conversion.

```
SELECT PARSE('Monday, 13 August 2012' AS datetime2 USING 'en-US') AS 'Date in English'
```

Date in English
2012-08-13 00:00:00.0000000

Section 43.4: Logical and Mathmetical Function

SQL has two logical functions – `CHOOSE` and `IIF`.

The `CHOOSE` function returns an item from a list of values, based on its position in the list. This position is specified by the index.

In the syntax, the index parameter specifies the item and is a whole number, or integer. The val_1 … val_n parameter identifies the list of values.

```
SELECT CHOOSE(2, 'Human Resources', 'Sales', 'Admin', 'Marketing' ) AS Result;
```

Result
Sales

In this example, you use the `CHOOSE` function to return the second entry in a list of departments.

The `IIF` function returns one of two values, based on a particular condition. If the condition is true, it will return true value. Otherwise it will return a false value.

In the syntax, the boolean_expression parameter specifies the Boolean expression. The true_value parameter specifies the value that should be returned if the boolean_expression evaluates to true and the false_value parameter specifies the value that should be returned if the boolean_expression evaluates to false.

```
SELECT BusinessEntityID, SalesYTD,
   IIF(SalesYTD > 200000, 'Bonus', 'No Bonus') AS 'Bonus?'
FROM Sales.SalesPerson
GO
```

BusinessEntityID	SalesYTD	Bonus?
274	559697.5639	Bonus
275	3763178.1787	Bonus
285	172524.4512	No Bonus

In this example, you use the IIF function to return one of two values. If a sales person's year-to-date sales are above 200,000, this person will be eligible for a bonus. Values below 200,000 mean that employees don't qualify for bonuses.

SQL includes several mathematical functions that you can use to perform calculations on input values and return numeric results.

One example is the `SIGN` function, which returns a value indicating the sign of an expression. The value of -1 indicates a negative expression, the value of +1 indicates a positive expression, and 0 indicates zero.

```
SELECT SIGN(-20) AS 'Sign'
```

Sign
-1

In the example, the input is a negative number, so the Results pane lists the result -1.

Another mathematical function is the `POWER` function. This function provides the value of an expression raised to a specified power.

In the syntax, the float_expression parameter specifies the expression, and the y parameter specifies the power to which you want to raise the expression.

```
SELECT POWER(50, 3) AS Result
```

Result
125000