**Stored Procedure and Function in SQL Server**

Stored Procedures are pre-compiled objects which are compiled for first time and its compiled format is saved which executes (compiled code) whenever it is called. But Function is compiled and executed every time when it is called.

**Basic Difference**

1. Function must return a value but in Stored Procedure it is optional (Procedure can return zero or n values).
2. Functions can have only input parameters whereas Procedures can have input/output parameters.
3. Function takes one input parameter, which is mandatory but Stored Procedure may take zero to n input parameters.
4. Functions can be called from Procedure whereas Procedures cannot be called from Function.

**Advanced Difference**

1. Procedure allows SELECT as well as DML (INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.
2. Procedures cannot be utilized in a SELECT statement whereas Function can be embedded in a SELECT statement.
3. Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be.
4. Functions that return tables can be treated as another rowset. This can be used in JOINs with other tables.
5. Inline Function can be thought of as Views that take parameters and can be used in JOINs and other rowset operations.
6. Exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.
7. We can go for Transaction Management in Procedure whereas we can't go in Function.

**Different Types of SQL Server Stored Procedures**

A stored procedure is a precompiled set of one or more SQL statements that is stored on SQL Server. Benefit of Stored Procedures is that they are executed on the server side and perform a set of actions, before returning the results to the client side. This allows a set of actions to be executed with minimum time and also reduce the network traffic. Hence stored procedure improves performance to execute SQL statements. For more about stored procedure refer the article [CRUD Operations using Stored Procedures](http://www.dotnet-tricks.com/Tutorial/sqlserver/I3W8010412-CRUD-Operations-using-Stored-Procedures.html).

Stored procedure can accept input and output parameters. Stored procedure can return multiple values using output parameters. Using stored procedure, we can Select, Insert, Update, and Delete data in database.

## Types of Stored Procedure

1. **System Stored Procedure**

These stored procedures are already defined in SQL Server. These are physically stored in hidden SQL Server Resource Database and logically appear in the sys schema of each user defined and system defined database. These procedure starts with the sp\_ prefix. Hence we don't use this prefix when naming user-defined procedures. Here is a list of some useful system defined procedure.

|  |  |
| --- | --- |
| System Procedure | Description |
| sp\_rename | It is used to rename an database object like stored procedures, views, tables etc. |
| sp\_changeowner | It is used to change the owner of a database object. |
| sp\_help | It provides details on any database object. |
| sp\_helpdb | It provides the details of the databases defined in the SQL Server. |
| sp\_helptext | It provides the text of a stored procedure reside in SQL Server |
| sp\_depends | It provides the details of all the database objects that depend on the specific database object. |

1. **Extended Procedure**

Extended procedures provide an interface to external programs for various maintenance activities. These extended procedures start with the xp\_ prefix and are stored in Master database. Basically these are used to call programs that reside on the server automatically from a stored procedure or a trigger run by the server.

**Example** Below statements are used to log an event in the NT event log of the server without raising any error on the client application.

* 1. **declare @logmsg varchar(100)**
  2. **set @logmsg = suser\_sname() + ': Tried to access the dotnet system.'**
  3. **exec xp\_logevent 50005, @logmsg**
  4. **print @logmsg**

**Example** The below procedure will display details about the BUILTIN\Administrators Windows group.

* 1. **EXEC xp\_logininfo 'BUILTIN\Administrators'**

1. **User-Defined Stored Procedure**

These procedures are created by user for own actions. These can be created in all system databases except the Resource database or in a user-defined database.

1. **CLR Stored Procedure**

CLR stored procedures are a special type of procedures that are based on the CLR (Common Language Runtime) in the .NET Framework. CLR integration of procedure was introduced with SQL Server 2008 and allow for procedure to be coded in any of the .NET languages like C#, Visual Basic and F#.

Note

1. We can nest stored procedures and managed code references in SQL Server up to 32 levels only. This is also applicable for function, trigger and view.
2. The current nesting level of a stored procedures execution is stored in the @@NESTLEVEL function.
3. In SQL Server stored procedure nesting limit is up to 32 levels, but there is no limit on the number of stored procedures that can be invoked with in a stored procedure

**Different Types of SQL Server Functions**

Function is a database object in SQL Server. Basically it is a set of SQL statements that accept only input parameters, perform actions and return the result. Function can return only single value or a table. We can’t use function to Insert, Update, and Delete records in the database table(s).

## Types of Functions

1. System Defined Functions

These functions are defined by SQL Server for different purpose. We have two types of system defined function in SQL Server

* 1. System Scalar Function

Scalar functions operate on a single value and returns a single value. Below is the list of some useful SQL Server Scalar functions.

|  |  |
| --- | --- |
| System Scalar Function | Description |
| abs(-10.67) | This returns absolute number of the given number means 10.67. |
| rand(10) | This will generate random number of 10 characters. |
| round(17.56719,3) | This will round off the given number to 3 places of decimal means 17.567 |
| upper('dotnet') | This will returns upper case of given string means 'DOTNET' |
| lower('DOTNET') | This will returns lower case of given string means 'dotnet' |
| ltrim(' dotnet') | This will remove the spaces from left hand side of 'dotnet' string. |
| convert(int, 15.56) | This will convert the given float value to integer means 15. |

* 1. System Aggregate Function

Aggregate functions operate on a collection of values and return a single value. Below is the list of some useful SQL Server Aggregate functions.

|  |  |
| --- | --- |
| System Aggregate Function | Description |
| max() | This returns maximum value from a collection of values. |
| min() | This returns minimum value from a collection of values. |
| avg() | This returns average of all values in a collection. |
| count() | This returns no of counts from a collection of values. |

1. User-Defined Functions

These functions are created by user in system database or in user defined database. We three types of user defined functions.

* 1. Scalar User-Defined Function

User defined scalar function also returns single value as a result of actions perform by function. We return any data type value from function.

* + 1. ***--Create a table***
    2. **CREATE TABLE Employee**
    3. **(**
    4. **EmpID int PRIMARY KEY,**
    5. **FirstName varchar(50) NULL,**
    6. **LastName varchar(50) NULL,**
    7. **Salary int NULL,**
    8. **Address varchar(100) NULL,**
    9. **)**
    10. ***--Insert Data***
    11. **Insert into Employee(EmpID,FirstName,LastName,Salary,Address) Values(1,'Mohan','Chauahn',22000,'Delhi');**
    12. **Insert into Employee(EmpID,FirstName,LastName,Salary,Address) Values(2,'Asif','Khan',15000,'Delhi');**
    13. **Insert into Employee(EmpID,FirstName,LastName,Salary,Address) Values(3,'Bhuvnesh','Shakya',19000,'Noida');**
    14. **Insert into Employee(EmpID,FirstName,LastName,Salary,Address) Values(4,'Deepak','Kumar',19000,'Noida');**
    15. ***--See created table***
    16. **Select \* from Employee**

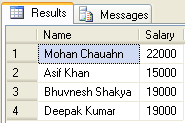


* + 1. ***--Create function to get emp full name***
    2. **Create function fnGetEmpFullName**
    3. **(**
    4. **@FirstName varchar(50),**
    5. **@LastName varchar(50)**
    6. **)**
    7. **returns varchar(101)**
    8. **As**
    9. **Begin return (Select @FirstName + ' '+ @LastName);**
    10. **end**



**1. *--Calling the above created function***

**2. Select dbo.fnGetEmpFullName(FirstName,LastName) as Name, Salary from Employee**



* 1. Inline Table-Valued User-Defined Function

User defined inline table-valued function returns a table variable as a result of actions perform by function. The value of table variable should be derived from a single SELECT statement.

* + 1. ***--Create function to get employees***
    2. **Create function fnGetEmployee()**
    3. **returns Table**
    4. **As**
    5. **return (Select \* from Employee)**



* + 1. ***--Now call the above created function***
    2. **Select \* from fnGetEmployee()**



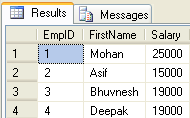
* 1. Multi-Statement Table-Valued User-Defined Function

User defined multi-statement table-valued function returns a table variable as a result of actions perform by function. In this a table variable must be explicitly declared and defined whose value can be derived from a multiple sql statements.

* + 1. ***--Create function for EmpID,FirstName and Salary of Employee***
    2. **Create function fnGetMulEmployee()**
    3. **returns @Emp Table**
    4. **(**
    5. **EmpID int,**
    6. **FirstName varchar(50),**
    7. **Salary int**
    8. **)**
    9. **As**
    10. **begin**
    11. **Insert @Emp Select e.EmpID,e.FirstName,e.Salary from Employee e;**
    12. ***--Now update salary of first employee***
    13. **update @Emp set Salary=25000 where EmpID=1;**
    14. ***--It will update only in @Emp table not in Original Employee table***
    15. **return**
    16. **end**



* + 1. ***--Now call the above created function***
    2. **Select \* from fnGetMulEmployee()**



* + 1. ***--Now see the original table. This is not affected by above function update command***
    2. **Select \* from Employee**



**Note**

1. Unlike Stored Procedures, Functions return only single value.
2. Unlike Stored Procedures, Functions accept only input parameters.
3. Unlike Stored Procedures, Functions are not used to Insert, Update, and Delete data in database table(s).
4. Like Stored Procedures, Functions can be nested up to 32 levels.
5. User-Defined Functions can have up to 1023 input parameters while Stored Procedures can have up to 2100 input parameters.
6. User-Defined Functions cannot return XML Data Type.
7. User-Defined Functions do not support Exception Handling.
8. User-Defined Function can call only Extended Stored Procedure.
9. User Defined Function doesn't support set options like set ROWCOUNT etc.

# Built-in Functions (Transact-SQL)

**SQL Server 2012**

[Other Versions](javascript:;)



* [SQL Server 2008 R2](http://technet.microsoft.com/en-us/library/ms174318(d=printer,v=sql.105).aspx)
* [SQL Server 2008](http://technet.microsoft.com/en-us/library/ms174318(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://technet.microsoft.com/en-us/library/ms174318(d=printer,v=sql.90).aspx)

SQL Server provides many built-in functions and also lets you create user-defined functions. The categories of built-in functions are listed on this page.

[Types of Functions](javascript:void(0))

|  |  |
| --- | --- |
| **Function** | **Description** |
| [Rowset Functions](http://technet.microsoft.com/en-us/library/ms187957.aspx) | Return an object that can be used like table references in an SQL statement. |
| [Aggregate Functions](http://technet.microsoft.com/en-us/library/ms173454.aspx) | Operate on a collection of values but return a single, summarizing value. |
| [Ranking Functions](http://technet.microsoft.com/en-us/library/ms189798.aspx) | Return a ranking value for each row in a partition. |
| Scalar Functions (Described below) | Operate on a single value and then return a single value. Scalar functions can be used wherever an expression is valid. |

[Scalar Functions](javascript:void(0))

|  |  |
| --- | --- |
| **Function category** | **Description** |
| [Configuration Functions](http://technet.microsoft.com/en-us/library/ms173823.aspx) | Return information about the current configuration. |
| [Conversion Functions](http://technet.microsoft.com/en-us/library/hh231076.aspx) | Support data type casting and converting. |
| [Cursor Functions](http://technet.microsoft.com/en-us/library/ms186285.aspx) | Return information about cursors. |
| [Date and Time Data Types and Functions](http://technet.microsoft.com/en-us/library/ms186724.aspx) | Perform operations on a date and time input values and return string, numeric, or date and time values. |
| [Logical Functions](http://technet.microsoft.com/en-us/library/hh213226.aspx) | Perform logical operations. |
| [Mathematical Functions](http://technet.microsoft.com/en-us/library/ms177516.aspx) | Perform calculations based on input values provided as parameters to the functions, and return numeric values. |
| [Metadata Functions](http://technet.microsoft.com/en-us/library/ms187812.aspx) | Return information about the database and database objects. |
| [Security Functions](http://technet.microsoft.com/en-us/library/ms186236.aspx) | Return information about users and roles. |
| [String Functions](http://technet.microsoft.com/en-us/library/ms181984.aspx) | Perform operations on a string (char or varchar) input value and return a string or numeric value. |
| [System Functions](http://technet.microsoft.com/en-us/library/ms187786.aspx) | Perform operations and return information about values, objects, and settings in an instance of SQL Server. |
| [System Statistical Functions](http://technet.microsoft.com/en-us/library/ms177520.aspx) | Return statistical information about the system. |
| [Text and Image Functions](http://technet.microsoft.com/en-us/library/ms188353.aspx) | Perform operations on text or image input values or columns, and return information about the value. |

[Function Determinism](javascript:void(0))

SQL Server built-in functions are either deterministic or nondeterministic. Functions are deterministic when they always return the same result any time they are called by using a specific set of input values. Functions are nondeterministic when they could return different results every time they are called, even with the same specific set of input values. For more information, see [Deterministic and Nondeterministic Functions](http://technet.microsoft.com/en-us/library/ms178091.aspx)

[Function Collation](javascript:void(0))

Functions that take a character string input and return a character string output use the collation of the input string for the output.

Functions that take non-character inputs and return a character string use the default collation of the current database for the output.

Functions that take multiple character string inputs and return a character string use the rules of collation precedence to set the collation of the output string. For more information, see [Collation Precedence (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms179886.aspx).

# Deterministic and Nondeterministic Functions

**SQL Server 2012**

[Other Versions](javascript:;)



* [SQL Server 2008 R2](http://technet.microsoft.com/en-us/library/ms178091(d=printer,v=sql.105).aspx)
* [SQL Server 2008](http://technet.microsoft.com/en-us/library/ms178091(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://technet.microsoft.com/en-us/library/ms178091(d=printer,v=sql.90).aspx)

Deterministic functions always return the same result any time they are called with a specific set of input values and given the same state of the database. Nondeterministic functions may return different results each time they are called with a specific set of input values even if the database state that they access remains the same.

There are several properties of user-defined functions that determine the ability of the SQL Server Database Engine to index the results of the function, either through indexes on computed columns that call the function, or through indexed views that reference the function. The determinism of a function is one such property. For example, a clustered index cannot be created on a view if the view references any nondeterministic functions. For more information about the properties of functions, including determinism, see [User-Defined Functions](http://technet.microsoft.com/en-us/library/ms191007.aspx).

This topic identifies the determinism of built-in system functions and the effect on the deterministic property of user-defined functions when it contains a call to extended stored procedures.

[Built-in Function Determinism](javascript:void(0))

You cannot influence the determinism of any built-in function. Each built-in function is deterministic or nondeterministic based on how the function is implemented by SQL Server.

All of the aggregate and string built-in functions are deterministic. For a list of these functions, see [Aggregate Functions (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms173454.aspx) and [String Functions (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms181984.aspx).

The following built-in functions from categories of built-in functions other than aggregate and string functions are always deterministic.

|  |  |  |
| --- | --- | --- |
| ABS | DATEDIFF | POWER |
| ACOS | DAY | RADIANS |
| ASIN | DEGREES | ROUND |
| ATAN | EXP | SIGN |
| ATN2 | FLOOR | SIN |
| CEILING | ISNULL | SQUARE |
| COALESCE | ISNUMERIC | SQRT |
| COS | LOG | TAN |
| COT | LOG10 | YEAR |
| DATALENGTH | MONTH |  |
| DATEADD | NULLIF |  |

The following functions are not always deterministic, but can be used in indexed views or indexes on computed columns when they are specified in a deterministic manner.

|  |  |
| --- | --- |
| **Function** | **Comments** |
| CAST | Deterministic unless used with datetime, smalldatetime, or sql\_variant. |
| CONVERT | Deterministic unless one of these conditions exists:   * Source type is sql\_variant. * Target type is sql\_variant and its source type is nondeterministic. * Source or target type is datetime or smalldatetime, the other source or target type is a character string, and a nondeterministic style is specified. To be deterministic, the style parameter must be a constant. Additionally, styles less than or equal to 100 are nondeterministic, except for styles 20 and 21. Styles greater than 100 are deterministic, except for styles 106, 107, 109 and 113. |
| CHECKSUM | Deterministic, with the exception of CHECKSUM(\*). |
| ISDATE | Deterministic only if used with the CONVERT function, the CONVERT style parameter is specified and style is not equal to 0, 100, 9, or 109. |
| RAND | RAND is deterministic only when a seed parameter is specified. |

All the configuration, cursor, metadata, security, and system statistical functions are nondeterministic. For a list of these functions, see [Configuration Functions (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms173823.aspx), [Cursor Functions (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms186285.aspx), [Metadata Functions (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms187812.aspx), [Security Functions (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms186236.aspx), and [System Statistical Functions (Transact-SQL)](http://technet.microsoft.com/en-us/library/ms177520.aspx).

The following built-in functions from other categories are always nondeterministic.

|  |  |
| --- | --- |
| @@CONNECTIONS | @@TOTAL\_WRITE |
| @@CPU\_BUSY | CURRENT\_TIMESTAMP |
| @@DBTS | GETDATE |
| @@IDLE | GETUTCDATE |
| @@IO\_BUSY | GET\_TRANSMISSION\_STATUS |
| @@MAX\_CONNECTIONS | MIN\_ACTIVE\_ROWVERSION |
| @@PACK\_RECEIVED | NEWID |
| @@PACK\_SENT | NEWSEQUENTIALID |
| @@PACKET\_ERRORS | NEXT VALUE FOR |
| @@TIMETICKS | PARSENAME |
| @@TOTAL\_ERRORS | RAND |
| @@TOTAL\_READ | TEXTPTR |

[Calling Extended Stored Procedures from Functions](javascript:void(0))

Functions that call extended stored procedures are nondeterministic, because the extended stored procedures can cause side effects on the database. Side effects are changes to a global state of the database, such as an update to a table, or to an external resource, such as a file or the network; for example, modifying a file or sending an e-mail message. You should not rely on returning a consistent result set when executing an extended stored procedure from a user-defined function. User-defined functions that create side effects on the database are not recommended.

When called from inside a function, the extended stored procedure cannot return result sets to the client. Any Open Data Services API that returns result sets to the client will have a return code of FAIL.

The extended stored procedure can connect back to SQL Server. However, the procedure cannot join the same transaction as the original function that invoked the extended stored procedure.

Similar to invocations from a batch or stored procedure, the extended stored procedure is executed in the context of the Microsoft Windows security account under which SQL Server is running. The owner of the extended stored procedure should consider this when granting permissions to other users to execute the procedure.