Overview of SQL Server Stored Procedure

Top of Form

**1. Overview**

When you need to store or retrieve data that is accessing and manipulating data within a database is a rather straightforward operation in SQL server database, and so we will go for T-SQL with four simple commands – SELECT, INSERT, DELETE, and UPDATE, and perform all the required tasks. The skill in working with databases lies in being able to not just work with these commands, but also complete each unit of work efficiently.

We will cover the different types of stored procedures used in SQL Server and their usage. We will also examine the working of stored procedures, the potentially tricky areas, and how to avoid them.

To increase performances, every developer should consider a number of factors during development. However, we will focus on running T-SQL from within stored procedures; we will examine what constitutes a stored procedure and discuss different types. We will cover:

* The difference between system stored procedures and user stored procedures
* Creating and designing stored procedures
* Using parameters and best practices
* Flow control statements
* Recursion
* Return values

**2. What is a Stored Procedure?**

If a repetitive T-SQL task has to be executed within an application, then the best repository for it is a program called a **stored procedure**, stored in SQL Server. Storing the code inside the SQL Server object gives us many advantages, like:

* Security due to encryption
* Performance gains due to compilation
* Being able to hold the code in a central repository:
  + Altering the code in SQL Server without replicating in several different programs
  + Being able to keep statistics on the code to keep it optimized
* Reduction in the amount of data passed over a network by keeping the code on the server
* Hiding the raw data by allowing only stored procedures to gain access to the data

You may have executed some ad-hoc queries for tasks like inserting data, querying information in other systems, or creating new database objects such as tables. All these tasks can be placed within a stored procedure, so that any developer can run the same code without having to recreate the T-SQL commands. Also, generalizing the code for all values makes it generic and reusable.

Stored procedures are more than just tools for performing repetitive tasks. There are two main types of stored procedure – system stored procedures and user-defined stored procedures. We also have extended stored procedures that can reside as either system or user-defined types. Extended stored procedures give functionality that is not necessarily contained within SQL Server, like allowing DOS commands to run and working with e-mail. It is also possible to create your own extended stored procedures.

Sample example of **stored procedure:**

*/\**

*DECLARE @OutPutValue VARCHAR(100)*

*EXEC spExample 'CodeProject', @OutPutValue OUTPUT*

*PRINT @OutPutValue*

*\*/*

CREATE PROCEDURE [dbo].[spExample]

@parameter1 VARCHAR(100)

,@parameter2 VARCHAR(200) OUTPUT

AS

BEGIN

DECLARE @parameter3 VARCHAR(100)

SET @parameter3 = ' Your development resources.'

IF @parameter1 IS NOT NULL AND LEN(@parameter1) > 1

SELECT @parameter2 = 'The '

+ @parameter1

+ @parameter3

ELSE SELECT @parameter2 = 'CodeProject is cool!'

RETURN

END

GO

More details on Integration of an OLE Object with SQL Server will be found at[this link](http://www.codeproject.com/KB/database/CLRExample_v2.aspx)**.**

**2.1 System Stored Procedures**

In SQL Server, many administrative and informational activities can be performed by using system stored procedures. Every time we add or modify a table, make a backup plan, or perform any other administrative function from within Enterprise Manager, we actually call a stored procedure specifically written to complete the desired action. These stored procedures are known as system stored procedures, which are functions that enhance the basic functionality of SQL Server itself, either by extending the functionality of an existing system stored procedure or by creating new functionality that enhances the basics of what is already there.

System stored procedures are prefixed by sp\_, so it is not advisable to use sp\_ for any of the stored procedures that we create, unless they form a part of our SQL Server installation. Creating a stored procedure prefixed with sp\_ and placing it in the master database will make it available to any database without the need to prefix the stored procedure with the name of the database. More details can be found at [this link](http://msdn.microsoft.com/en-us/library/ms187961.aspx).

Let's clarify this with an example. If we take the sp\_who stored procedure, call it sp\_mywho, store it in the masterdatabase, and move to another database such as northwind, we can still **execute**sp\_mywho, rather than having to specify the procedure in the fully qualified manner as master.dbo.sp\_mywho.

**2.2 User Stored Procedures**

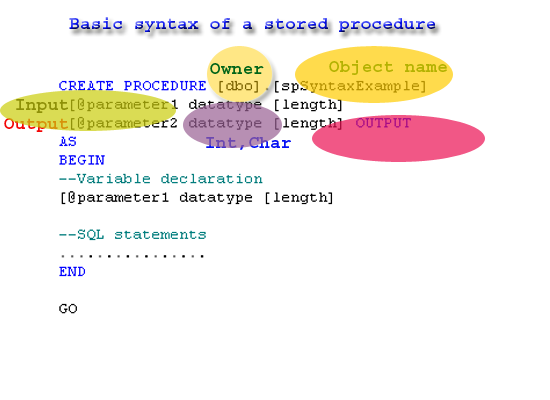
A user stored procedure is any program that is stored and compiled within SQL Server (but not in the master database) and prefixed with sp\_. User stored procedures can be categorized into three distinct types:

* User stored procedures
* Triggers, and
* User defined functions

**2.3 Creating Stored Procedures**

The creation process depends on what we want it to do, now let's take a look at the syntax for creating a stored procedure:

**Syntax**



An example of a simple stored procedure follows, where two numbers are passed in and the midpoint of the two numbers is listed:

CREATE PROCEDURE ut\_MidPoint @LowerNumber int, @HigherNumber int

AS

BEGIN

DECLARE @Mid int

IF @LowerNumber > @HigherNumber

RAISERROR('You have entered your numbers the wrong way round',16,1)

SET @Mid = ((@HigherNumber - @LowerNumber) / 2) + @LowerNumber

SELECT @Mid

END

At the time of creation, SQL Server takes our code and parses it for any syntactical errors. Column names and variables are checked for existence at compilation. Even if they don't exist, any temporary tables created within the stored procedure will also pass the compilation stage. This is known as **deferred name resolution**. It can be an asset or a drawback, as we can create temporary tables that exist for the lifetime of the stored procedure execution only, which is desirable, but if we define a temporary table and get something wrong later on, such as a column name, then the compilation will not pick up this error.

Once it is compiled, the details of the stored procedure are stored in three system tables in the concerned database, they are as follows:

sysobjects

Contains one row for each object (constraint, default, log, rule, stored procedure, and so on) created within a database. In **tempdb** only, this table includes a row for each temporary object.

More details can be found at [this link](http://msdn.microsoft.com/en-us/library/aa260447%28SQL.80%29.aspx).

sysdepends

Contains dependency information between objects (views, procedures, and triggers) in the database, and the objects (tables, views, and procedures) that are contained in their definition.

More detail can be found at [this link](http://msdn.microsoft.com/en-us/library/ms190325.aspx).

syscomments

Contains entries for each view, rule, default, trigger, CHECK constraint, DEFAULT constraint, and stored procedure within the database. The **text** column contains the original SQL definition statements.

More details can be found at[this link](http://msdn.microsoft.com/en-us/library/ms186293.aspx).

**2.3.1 Performance Consideration**

When a stored procedure is created, it passes through several steps. First of all, the T-SQL is parsed and resolved, saved to the disk, and stored in SQL Server. The first time the procedure is executed, the procedure is retrieved and optimized, on the basis of any data queries and values passed through parameters. SQL Server will inspect the code and try to use the best indexes on the tables, which are referenced by checking the statistics that are held for those tables.

The query plan is then cached within SQL Server, ready for any further executions. SQL Server will always use this plan, providing it doesn't retire the plan. Thus, the performance gain of stored procedures comes from compiled cached plans.

2.3.2 Network Consideration

You may consider passing T-SQL statements to insert a row into a table, with very few characters. However, creating a stored procedure and passing only the name of the stored procedure, parameters, and their values reduces the characters needed. We can see the extra overhead imposed by the T-SQL statements on our network by multiplying this difference by the number of calls being made. This can be a significant issue, especially if we are using it over the Internet.

Compare the next two statements with each other:

CREATE PROCEDURE ut\_MidPoint @LowerNumber int, @HigherNumber int

AS

BEGIN

DECLARE @Mid int

IF @LowerNumber > @HigherNumber

RAISERROR('You have entered your numbers the wrong way round',16,1)

SET @Mid = ((@HigherNumber - @LowerNumber) / 2) + @LowerNumber

SELECT @Mid

END

The first statement has 74 characters, while the second has 46 characters, differing by a mere 28 characters. However, if this was a more column-intensive insert, with 10,000 of these in a day, for example, this amounts to almost 280k of wasted bandwidth! What if an image data type was being uploaded or downloaded? Anything that is of binary data type, such as images or sounds, and so on, is sent as binary values. These are converted to character strings, and this will double the size of the ad-hoc query that we are sending, when using T-SQL inline.

**3. How Stored Procedures Work**

When we run a stored procedure, Adaptive Server prepares an execution plan so that the procedure's execution is very fast. Stored procedures can:

* Take parameters
* Call other procedures
* Return a status value to a calling procedure or batch to indicate success or failure and the reason for failure
* Return values of parameters to a calling procedure or batch
* Be executed on remote Adaptive Servers

The ability to write smart stored procedures significantly enhances the power, efficiency, and flexibility of SQL. Compiled procedures radically improve the performance of SQL statements and batches. In addition, stored procedures on other Adaptive Servers can be executed if both your server and the remote server are set up to allow remote logins. You can write triggers on your local Adaptive Server that executes procedures on a remote server whenever certain events, such as deletions, updates, or inserts, take place locally.

Stored procedures differ from ordinary SQL statements and from batches of SQL statements in that they are precompiled. The first time you run a procedure, Adaptive Server's query processor analyzes it and prepares an execution plan that is ultimately stored in a system table. Subsequently, the procedure is executed according to the stored plan. Since most of the query processing work has already been performed, stored procedures execute almost instantly.

**4. Parameter Usage**

Data will be passed in and out of a stored procedure, even when the purpose of the procedure is not to return information. All variables defined as parameters have to be prefixed with an @ sign. The normal practice is to define input parameters before any output parameters.

As with any column in a database, use the data type that is most appropriate for the parameter and if the parameter has to match with a column in the table, then ensure that both the data types match exactly.

If the parameter does not match a column but will be still used, for example, as a join condition or a filter condition, then use a data type and size appropriate to the type of data, rather than have everything as character data type. Also, wherever possible, use varchar or nvarchar rather than char, just as you will in your inline T-SQL, which will avoid unnecessary spaces being passed across networks.

**4. Calling a Stored Procedure**

Many people tend to miss a performance enhancement related to executing or calling a stored procedure. For example, if you wanted to call the [Ten Most Expensive Products] stored procedure in thenorthwind database, you can simply do it as:

[Ten Most Expensive Products]

You can skip EXEC(UTE), but you will need it if you run one stored procedure within another.

However, this is not the most efficient way to call a stored procedure. When such a command is processed, SQL Server has a hierarchical method of finding a stored procedure and executing it. First of all, it will take the login ID of the user who wants to execute the procedure and see if a stored procedure of that name exists. If it doesn't, then SQL Server will look for the stored procedure under the DBO login. Hence, always fully qualify your stored procedure with the owner.

**Summary**

We covered several issues concerning stored procedures. We have looked at what a stored procedure is, how to create one, and how stored procedures can ensure your development runs at its optimum. Even when a database is being replicated, stored procedures can and do improve performance.

**Points of Interest**

Don't just forget your stored procedures once they are created, just as you will never forget your indexes. When a new index is added to a table, or any sort of modification is done to any dependent objects, the whole query tree requires compilation.

Finally, don't make stored procedures what they are not. Don't replace SQL Server functionality such as CHECK constraints with a stored procedure unless the CHECK constraint doesn't offer enough functionality

Bottom of Form