

In This Issue

- **An overview of the Microsoft MVP program.**
- **Wally does a performance test to compare TSql and SQL CLR functions**

Microsoft MVPs

By Luci Harrell

Though we're all probably familiar with the acronym 'MVP,' many might not know the implications of it as it relates to developers in the .NET world. For Microsoft, MVP stands for 'Most Valuable Professional,' and is an award program that the company's been behind for over 10 years.

Microsoft gives MVP awards to technology professionals who exhibit ingenuity in their respective fields and share their expertise with other developers. They look for individuals who go above and beyond the scope of their own jobs, offering training, insight and involvement as leaders in their communities.

Scalable Development partner, Wally McClure, just received his fifth MVP award. His is in the area of ASP.NET, but all are not. In fact, MVPs are technical pros who, altogether, work with over 90 different types of Microsoft driven technologies. Several of the authors with which Scalable Development partners have written technical articles and programming books are also Microsoft MVPs.

This honor is important to awardees because they are chosen based on input from a worldwide community of programming peers. There are only about 800 development MVPs, representing over 90 countries around the world.

For more information about the Microsoft MVP program, visit www.mvp.support.microsoft.com. ■

TSql vs. SQL CLR Performance Analysis

By Wallace "Wally" McClure

About a year ago I spoke to the Michiana .NET User Group in South Bend, IN. The subject of the talk was Sql Server 2005 CLR Objects. The question at hand: Which should I use - TSql or CLR objects? Personally, my answer has always been this: if you can do it in TSql, it's best to do so. If you can perform the necessary operations using TSql, in general, the performance will be better. This is before one takes into account the project overhead of managing the source code for stored procedures, triggers and other database objects.

Once you decide that your scenario actually needs a CLR Object, what's the best way to perform the operation? Should this be done entirely within a CLR Object? I have decided to take a look at these questions based on the WebSearch spider and its use of CLR Objects to perform certain data operations.

Question: What is the best way to implement a CLR Object within a Sql Server 2005 Database? Which will perform the best?¹

Short Answer: Do as much as you possibly can in TSql. Use CLR Objects judiciously. Once the need is determined, perform as many operations in TSql and use CLR Objects only as absolutely needed.

Much Longer Answer: First off, not all situations are the same. In the following example, we are looking at this from the basic standpoint of keeping data synchronized within a single database table. To answer this question, I have built four different solutions to this problem:

See *Analysis* on Page 2

¹ Terms like "best" are highly subjective. As a result, the question that was answered is "Which performs the best?" Obviously, that type of question is very dependent on the scenario. As a result, please test this in your own shop to get the best results for your specific application.

Analysis

- Mostly TSql – A TSQL trigger which creates a cursor and calls out to some custom CLR functions on an individual basis.
- TSql Cursor – A TSQL trigger that creates a cursor that calls out to some custom CLR functions by looking within the trigger.
- A CLR Trigger which uses a Generic List <String> to hold the pending data along with a SqlDataReader to iterate through data. Each url is updated through the Generic List.
- A CLR Trigger which uses a DataSet/DataTable to hold the data and then the Sql Data Adapter's .Update method to transfer the data back.

	Average time	Standard Deviation
Mostly TSql (base)	1x	10 / 1x
TSql Cursor	1.42x	.61x
CLR with Generic List	2.04x	10.4x
CLR with Datasets	4.65x	15.6x

Performance Analysis²

Now that we have seen the performance differences of the four solutions, let's try and analyze the differences between them.

The Mostly TSql solution performs the best. This is not a surprise. TSql has been around for a number of years, has been embedded within SQL Server and has a highly optimized relationship with the database. When embedding the CLR with the database, it provides its own overhead, so it's not a shock that this solution performs the best.

The TSql Cursor solution takes 1.42 times as long as the Mostly TSql solution to perform the same work. This is not surprising; cursors generally take longer to perform a given operation than comparable sql commands.

The real surprise in this test is the performance of the CLR trigger, which uses a Generic List to hold its sql commands. The Generic List Trigger performed more than twice as fast as the Dataset Trigger. Personally, I would expect the Dataset to be more optimized, however, it does not perform as well. This is probably due to the fact that it creates much more overhead than a 'simpler' Generic List.

How was the data created? It is important to communicate how the above data was calculated.

1. In each situation, approximately 15 updates were performed against the 1000 records in the table. The command is "UPDATE TBLSEARCHRESULTS SET SERVERNAME=NULL"
2. When testing was performed, the triggers that were not being tested were not installed on the database table.
3. The first five updates were not used for the calculation. This is due to the startup and JIT operations that are necessities for the objects within the database. The idea is to simulate a running system. As a result, startup isn't a major concern for this test.
4. The sql commands AVG() and STDEV() were used to create the data in the above table.

The test platform for the scenario was as follows: a laptop with a 1.8 GHz Pentium 4 processor, 1 gigabyte of RAM, 60 gigabyte hard drive, Windows 2003 Server with Service Pak 1 and up-to-date security updates through Windows Update and Sql Server 2005 Service Pak 1.

There are a number of issues and assumptions made within this test that could've caused problems with the data that was generated. Assumptions made include:

- There is no need to go outside of the single database. If there was a need to go outside of the database, this test would not be applicable to your needs.
- Only updates were tracked. No inserts were tracked.
- This was tested using a laptop, not a multi-processor database server with an appropriate drive system layout. Different results for different hardware layout would not be unexpected.
- I don't believe I have any Sql Injection attack openings, but I might be wrong. ■

Scalable Development, Inc.

Atlanta Office

2900 Chamblee Tucker Rd.
Building 5
Atlanta, GA 30341
770-458-6590

Knoxville Office

118 Durwood Rd.
Knoxville, TN 37922
865-693-3004

www.scalabledevelopment.com

² Original source code for this test can be found at www.morewally.com.