MSSQLTips.com
brought to you by Edgewood Solutions

Your daily source for SQL Server Tips

Dynamic SQL execution on remote SQL Server using EXEC AT

Written By: Arshad Ali -- 5/22/2009

Problem
With SQL Server 2000, we had OPENQUERY and OPENROWSET to execute a pass-through query on the specified server, but it has several inherent limitations. Starting with SQL Server 2005 we have another more elegant way using "EXEC AT" to execute a pass-through query on the specified linked server which also addresses several shortcomings of OPENQUERY and OPENROWSET table functions.

In this tip I am going to start my brief discussion with OPENQUERY and OPENROWSET table functions, its limitation and how the new EXEC AT command overcomes them.

Solution
OPENQUERY table function executes the specified pass-through query on the specified linked server. This server is an OLE DB data source. OPENQUERY can be referenced in the FROM clause of a query as if it were a table name. OPENQUERY can also be referenced as the target table of an INSERT, UPDATE, or DELETE statement. Although the query may return multiple result sets, OPENQUERY returns only the first one.

The problems with OPENQUERY are; first, OPENQUERY does not accept variables for its arguments, in other words it must be static (although here is a trick to overcome it), second, only one result-set is returned and third, OPENQUERY is used in the FROM clause which is quite limiting in terms when you have to invoke an executable statement, such as a DDL statement against the target server. Similar limitation apply to the OPENROWSET table function as well.

SQL Server 2005 introduces an enhancement to the EXEC command to allow dynamic SQL execution on the linked server. The new EXEC AT command addresses the above limitations of OPENQUERY and OPENROWSET. EXEC AT specifies that command_string is executed against linked_server_name and results, if any, are returned to the client. The linked_server_name must refer to an existing linked server definition in the local server.

Example
So now let's see how we can use it. In the below given table,

- Script 1 creates a linked server definition for FARAWAYSERVER server.
- Script 2 executes a simple SELECT statement on the linked server which returns a single result-set.
- Script 3 executes two SELECT statements on the linked server and hence as result of this two result-sets are returned to the client.
- Script 4 executes a SELECT statement on the linked server and passes two arguments dynamically at run-time.
- Script 5 uses script 4, but this time it uses variables to pass argument values dynamically at run-time.
- Script 6 executes the executable code on the linked server, in this case its creating a table in tempdb on the linked server. And finally
- Script 7 drops all the objects created in this session.

Script : EXEC AT Command

```
--Script 1 : Create a linked server
EXEC sp_addlinkedserver 'FARAWAYSERVER', 'SQL Server'
--Script 2 : Execute a simple SELECT statement on the linked server
EXEC ('SELECT TOP 10 * FROM AdventureWorksLT.SalesLT.Customer') AT [FARAWAYSERVER];
GO

--Script 3 : Executing multiple SELECT statements on linked server and getting multiple resultsets
EXEC ('SELECT TOP 10 * FROM AdventureWorksLT.SalesLT.Customer;
SELECT TOP 10 * FROM AdventureWorksLT.SalesLT.CustomerAddress;') AT [FARAWAYSERVER];
GO
--Script 4 : Execute a SELECT statement on linked serer and pass two arguments at dynamically
EXEC ('SELECT TOP 10 * FROM AdventureWorksLT.SalesLT.Customer
WHERE CustomerID = ? AND LastName = ?', 10, 'Garza') AT [FARAWAYSERVER];
GO
--Script 5 : Execute a SELECT statement on linked serer and pass two arguments at dynamically
```

```
--by using variables
DECLARE @CustomerID AS INT
DECLARE @LastName AS VARCHAR(100)
SET @CustomerID = 10
SET @LastName = 'Garza'
EXEC ('SELECT TOP 10 * FROM AdventureWorksLT.SalesLT.Customer
WHERE CustomerID = ? AND LastName = ?', @CustomerID, @LastName) AT [FARAWAYSERVER];
GO
--Script 6 : Execute a DDL statement on linked server
EXEC (
'USE TempDB
IF OBJECT_ID(''dbo.Table1'') IS NOT NULL
DROP TABLE dbo.Table1
CREATE TABLE dbo.Table1
(
Column1 INT
)' ) AT [FARAWAYSERVER];
--Script 7 : Once you are done with your testing, clean up created objects
EXEC (
'USE TempDB
IF OBJECT_ID(''dbo.Table1'') IS NOT NULL
DROP TABLE dbo.Table1'
) AT [FARAWAYSERVER];
EXEC sp_dropserver 'FARAWAYSERVER'
```

Notes

- You may get error "Server '*myserver*' is not configured for RPC." for that you will not need to enable RPC with the commands given below:

  ```
  exec sp_serveroption @server='myserver', @optname='rpc', @optvalue='true'
  exec sp_serveroption @server='myserver', @optname='rpc out', @optvalue='true'
  ```

- Before you call EXECUTE with a character string, validate the character string. Never execute a command constructed from user input that has not been validated. For more information, see SQL Injection.
- If you have a named instance you will probably need to use sp_addlnkedserver as follows to create the linked server name without a slash. So the linked server is referenced as "SQL2005", but this will connect to instance "SERVER1\SQL2005".

  ```
  EXEC sp_addlinkedserver @server='SQL2005', @srvproduct='', @provider='SQLNCLI',
  @datasrc='SERVER1\SQL2005'
  ```

Next Steps

- Review the usage of sp_addlinkedserver for additional options.
- Review the usage of EXEC statement.
- Review Execute Dynamic SQL commands in SQL Server tip.
- Review Microsoft Access Pass-Through Queries to SQL Server tip.
- Review Using OPENROWSET to read large files into SQL Server tip.
- Review Granting permission with the EXECUTE AS command in SQL Server 2005 tip.
- Review Switching Stored Procedure Execution Context in SQL Server using the REVERT clause tip.