

How to Compare the Columns of Two SQL Server Tables

By Rick Obsitnik



About the Author

Rick, a specialist in database application development, who recently made the change to database administration, has worked in IT for 25 years in nearly all areas excluding sales! Quite an achievement! He currently works in a SQL Server 2005, ESRI ArcGIS SDE 9.3 environment with Visual Studio 2005. His appetite for knowledge in all things IT grows daily, although his first love is databases. His interest in humor and the absurd led to the creation of the [Sniglets](#) section on the SQL Server Club web site.

You can read more articles and blogs by Rick on his [Critical Status blog](#), which was designed to keep track of and access his database and development related thoughts, ideas and issues.

Over the past year or so, I have been asked to use a 'script' for comparing columns in two tables. 'You know the one in SQL!' Huh? I say, there isn't one, and then get weird looks.

So after I was recently asked and had someone do it manually, I wrote a stored procedure that will compare the general metadata of two tables across any database and any server.

It's not an elegant solution, but it will tell you what table a unique column falls in and what its data type and size is and if it's nullable. There's room for plenty of improvement on what it returns but the core is functional.

One of the keys to doing this across an enterprise is to use linked servers. You'll need to add a linked server in the instance Server Objects\Linked Servers. Linked servers are rather useful in that you can query across instances/servers. I use it frequently to run SELECTS between GIS databases and non-spatial web application databases. It's also useful for querying databases that exist in multiple environments (production, user test and development per se).

Also necessary is the ability to use the USE command to access beyond the one the stored proc resides in. This consists of building a command in a NVARCHAR generally set to a high size like 1000-4000 characters. The command will be run with the EXEC command. Example:

```
SELECT @COMMAND =  
    USE [' + @LINKEDSERVER + '].[' + @DBNAME + '];  
    SELECT  
        *  
    FROM
```

```

        [' + @LINKEDSERVER + '].[' + @DBNAME + '].[' + @TABLE +
        ];
    EXEC(@COMMAND);

```

This alone is some useful code to do some things you wished you could do. ;-)

In essence, to do the compare, you need to query the [SERVER].[DATABASE].[INFORMATION_SCHEMA].[COLUMNS] table for the two tables. You'll need to do an EXCEPT query of a SELECT of table one's columns data against the second table and an EXCEPT query of a SELECT of the second table one's sysobjects data against the first table. You can swap the order of the tables depending on which table is your primary and which is being searched against.

If you combine the two EXCEPT queries with a UNION, it will give you the same results but you can not easily determine which table the uniqueness can be found. Hence, I break it up into two queries, two processes.

PRINT commands can display the result of these queries or you can save them to a table etc. As speed is not of the essence in small table querying, I use cursors to run in memory which can then be disposed of. It's not very complex code, but it is useful. I can tell you users/co-workers will be grateful for such a simple solution.

This can be a handy tool in preparation for imports to verify the equivalency of source data metadata. Processes can fail due to a change in a source database. If data is pulled from a backup and there has been an upgrade or user defined tables have changed, this can get you started on the road to repair. I'm sure you data shepherds can find a variety of reasons for comparing metadata. Share your experiences in this former conundrum!

The stored procedure currently is running on SQL Server 2005 Enterprise Edition Service Pack 2. As I do not have a 2000 nor 2008 instance running, it is assumed that it will run in both environments. Below is the code to create the stored procedure:

```

__*****
__*****
__*****

USE [DB]
GO
/***** Object:  StoredProcedure [dbo].[spCOMPARETABLECOLUMNS]    Script Date: 06/01/2009
13:00:56 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
--
CREATE PROCEDURE [dbo].[spCOMPARETABLECOLUMNS]
    -- Add the parameters for the stored procedure here
    @INSTANCE1 NVARCHAR(35),
    @DATABASE1 NVARCHAR(35),
    @TABLE1 NVARCHAR(100),
    @INSTANCE2 NVARCHAR(35),
    @DATABASE2 NVARCHAR(35),
    @TABLE2 NVARCHAR(100)

```

AS

BEGIN

```
DECLARE @COMMAND NVARCHAR(4000),
        @COLUMNS_TABLE1 NVARCHAR(200),
        @COLUMNS_TABLE2 NVARCHAR(200),
        @NAME NVARCHAR(50),
        @IS_NULLABLE VARCHAR(3),
        @DATA_TYPE NVARCHAR(128),
        @CHARACTER_MAXIMUM_LENGTH INT,
        @NUMERIC_PRECISION INT,
        @NUMERIC_PRECISION_RADIX INT,
        @NUMERIC_SCALE TINYINT,
        @DATETIME_PRECISION SMALLINT,
        @TABLE NVARCHAR(50),
        @COUNT INT

SET @COUNT = 1
SET @INSTANCE1 =
    CASE
        WHEN (@INSTANCE1 IS NOT NULL) THEN '[' + RTRIM(@INSTANCE1) + ']. '
        WHEN (@INSTANCE1 IS NULL) THEN ''
    END

SET @DATABASE1 =
    CASE
        WHEN (@DATABASE1 IS NOT NULL) THEN '[' + RTRIM(@DATABASE1) + ']. '
        WHEN (@DATABASE1 IS NULL) THEN ''
    END

SET @INSTANCE2 =
    CASE
        WHEN (@INSTANCE2 IS NOT NULL) THEN '[' + RTRIM(@INSTANCE2) + ']. '
        WHEN (@INSTANCE2 IS NULL) THEN ''
    END

SET @DATABASE2 =
    CASE
        WHEN (@DATABASE2 IS NOT NULL) THEN '[' + RTRIM(@DATABASE2) + ']. '
        WHEN (@DATABASE2 IS NULL) THEN ''
    END

SET @COLUMNS_TABLE1 = RTRIM(@INSTANCE1) + RTRIM(@DATABASE1) +
'[INFORMATION_SCHEMA].[COLUMNS] '
SET @COLUMNS_TABLE2 = RTRIM(@INSTANCE2) + RTRIM(@DATABASE2) +
'[INFORMATION_SCHEMA].[COLUMNS] '

PRINT 'WHAT IS DIFFERENT BETWEEN ' + @TABLE1 + ' AND ' + @TABLE2
PRINT ''
PRINT ''

SELECT @COMMAND =
    'DECLARE c CURSOR FOR
    SELECT
        COLUMN_NAME,
        DATA_TYPE,
        CHARACTER_MAXIMUM_LENGTH,
        NUMERIC_PRECISION,
        NUMERIC_PRECISION_RADIX,
        NUMERIC_SCALE,
        DATETIME_PRECISION,
        IS_NULLABLE
    FROM '
    + @COLUMNS_TABLE1 + '
    WHERE
        TABLE_NAME = ' + CHAR(39) + @TABLE1 + CHAR(39) + '
```

```

EXCEPT
SELECT
    COLUMN_NAME,
    DATA_TYPE,
    CHARACTER_MAXIMUM_LENGTH,
    NUMERIC_PRECISION,
    NUMERIC_PRECISION_RADIX,
    NUMERIC_SCALE,
    DATETIME_PRECISION,
    IS_NULLABLE
FROM '
    + @COLUMNS_TABLE2 + '
WHERE
    TABLE_NAME = ' + CHAR(39) + @TABLE2 + CHAR(39)

BEGIN TRY
    EXEC(@COMMAND);
END TRY

BEGIN CATCH
    PRINT 'ERROR_MESSAGE = ' + ERROR_MESSAGE() + ' ERROR LINE = ' +
STR(ERROR_LINE())
    PRINT ''

END CATCH

OPEN c
FETCH NEXT FROM
    c
INTO
    @NAME,
    @DATA_TYPE,
    @CHARACTER_MAXIMUM_LENGTH,
    @NUMERIC_PRECISION,
    @NUMERIC_PRECISION_RADIX,
    @NUMERIC_SCALE,
    @DATETIME_PRECISION,
    @IS_NULLABLE

WHILE @@FETCH_STATUS = 0
    BEGIN
        BEGIN TRY
            PRINT 'FETCH ' + STR(@COUNT)
            SET @COUNT = @COUNT + 1
            PRINT 'COLUMN: ' +
                @NAME + ', ' +
                @DATA_TYPE + ', ' +
                CASE
                    WHEN @CHARACTER_MAXIMUM_LENGTH IS NULL
                THEN '0'
                    ELSE CAST(@CHARACTER_MAXIMUM_LENGTH AS
NVARCHAR(10))
                END
                + ', ' +
                CASE
                    WHEN @NUMERIC_PRECISION IS NULL THEN '0'
                    ELSE CAST(@NUMERIC_PRECISION AS
NVARCHAR(10))
                END
                + ', ' +

```

```

CASE
    WHEN @NUMERIC_PRECISION_RADIX IS NULL THEN
        '0'
    ELSE CAST(@NUMERIC_PRECISION_RADIX AS
        NVARCHAR(10))
END
+ ', ' +
CASE
    WHEN @NUMERIC_SCALE IS NULL THEN '0'
    ELSE CAST(@NUMERIC_SCALE AS NVARCHAR(10))
END
+ ', ' +
CASE
    WHEN @DATETIME_PRECISION IS NULL THEN '0'
    ELSE CAST(@DATETIME_PRECISION AS
        NVARCHAR(10))
END
+ ', ' +
@IS_NULLABLE
PRINT "

END TRY

BEGIN CATCH

    PRINT 'ERROR: ' + ERROR_MESSAGE()
    PRINT "

END CATCH

FETCH NEXT FROM
    c
INTO
    @NAME,
    @DATA_TYPE,
    @CHARACTER_MAXIMUM_LENGTH,
    @NUMERIC_PRECISION,
    @NUMERIC_PRECISION_RADIX,
    @NUMERIC_SCALE,
    @DATETIME_PRECISION,
    @IS_NULLABLE

END

CLOSE c
DEALLOCATE c

PRINT 'WHAT IS DIFFERENT BETWEEN ' + @TABLE2 + ' AND ' + @TABLE1
PRINT "
PRINT "

SELECT @COMMAND =
    'DECLARE c CURSOR FOR
        SELECT
            COLUMN_NAME,
            DATA_TYPE,
            CHARACTER_MAXIMUM_LENGTH,
            NUMERIC_PRECISION,
            NUMERIC_PRECISION_RADIX,
            NUMERIC_SCALE,
            DATETIME_PRECISION,
            IS_NULLABLE
        FROM '
        + @COLUMNS_TABLE2 + '
        WHERE
            TABLE_NAME = ' + CHAR(39) + @TABLE2 + CHAR(39) + '
    '

```

```

EXCEPT
SELECT
    COLUMN_NAME,
    DATA_TYPE,
    CHARACTER_MAXIMUM_LENGTH,
    NUMERIC_PRECISION,
    NUMERIC_PRECISION_RADIX,
    NUMERIC_SCALE,
    DATETIME_PRECISION,
    IS_NULLABLE
FROM '
    + @COLUMNS_TABLE1 + '
WHERE
    TABLE_NAME = ' + CHAR(39) + @TABLE1 + CHAR(39)

BEGIN TRY
    EXEC(@COMMAND);
END TRY
BEGIN CATCH
    PRINT 'ERROR_MESSAGE = ' + ERROR_MESSAGE() + ' ERROR LINE = ' +
STR(ERROR_LINE())
    PRINT ''

END CATCH

OPEN c
FETCH NEXT FROM
    c
INTO
    @NAME,
    @DATA_TYPE,
    @CHARACTER_MAXIMUM_LENGTH,
    @NUMERIC_PRECISION,
    @NUMERIC_PRECISION_RADIX,
    @NUMERIC_SCALE,
    @DATETIME_PRECISION,
    @IS_NULLABLE

SET @COUNT = 1

WHILE @@FETCH_STATUS = 0
    BEGIN
        BEGIN TRY
            PRINT 'FETCH ' + STR(@COUNT)
            SET @COUNT = @COUNT + 1
            PRINT 'COLUMN: ' +
                @NAME + ', ' +
                @DATA_TYPE + ', ' +
                CASE
                    WHEN @CHARACTER_MAXIMUM_LENGTH IS NULL
THEN '0'
                    ELSE CAST(@CHARACTER_MAXIMUM_LENGTH AS
NVARCHAR(10))
                END
                + ', ' +
                CASE
                    WHEN @NUMERIC_PRECISION IS NULL THEN '0'
                    ELSE CAST(@NUMERIC_PRECISION AS
NVARCHAR(10))
                END

```

```

        END
        + ',' +
CASE
        WHEN @NUMERIC_PRECISION_RADIX IS NULL THEN
'0'
        ELSE CAST(@NUMERIC_PRECISION_RADIX AS
NVARCHAR(10))
    END
    + ',' +
CASE
        WHEN @NUMERIC_SCALE IS NULL THEN '0'
        ELSE CAST(@NUMERIC_SCALE AS NVARCHAR(10))
    END
    + ',' +
CASE
        WHEN @DATETIME_PRECISION IS NULL THEN '0'
        ELSE CAST(@DATETIME_PRECISION AS
NVARCHAR(10))
    END
    + ',' +
@IS_NULLABLE
PRINT "

END TRY

BEGIN CATCH

    PRINT 'ERROR: ' + ERROR_MESSAGE()

END CATCH

FETCH NEXT FROM
    c
INTO
    @NAME,
    @DATA_TYPE,
    @CHARACTER_MAXIMUM_LENGTH,
    @NUMERIC_PRECISION,
    @NUMERIC_PRECISION_RADIX,
    @NUMERIC_SCALE,
    @DATETIME_PRECISION,
    @IS_NULLABLE

END

CLOSE c
DEALLOCATE c

```

So to sum it up, you'll always find requests to "run that script". People will want to compare whether two seemingly equivalent data sources are different. They will always go to their friendly data shepherd to make it happen. This little diddy will give them what they want and reduce the extra work load that comes with each request. It's good to be there hero while exerting so little effort to do so ;-)