**A script to answer: Should I use SPARSE?**

**By** [**Jeff Reinhard**](http://www.sqlservercentral.com/Authors/Scripts/Jeff_Reinhard/6404/)**, 2015/04/22**

**FROM:** [**http://www.sqlservercentral.com/scripts/sparse/124784/**](http://www.sqlservercentral.com/scripts/sparse/124784/)

I recently was exploring some options to redesign tables that had many columns; most of the columns were NULL for most rows. The issue is the table growth size is reducing scalability and performance.

Some of the options I considered were: Named attribute value pairs, Serialization, XML, and Vertical Fragmentation. All of these options had pros and cons, but the impact on the application layer is an important consideration.

Then I ran across the SPARSE option, first available in SQL 2008. In summary: *Sparse columns are ordinary columns that have an optimized storage for null values. Sparse columns reduce the space requirements for null values at the cost of more overhead to retrieve nonnull values.* This is a great option to improve scalability without changing the physical data model.

The usage is easy, either create the column or alter like this example:

ProductionSpecification varchar(20) SPARSE NULL,

The script is a stored procedure that takes a schema name and table name and it will report back on each of the columns, how many rows are null, the percentage of rows that are null, the percentage of null threshold for the data type of the column and if sparse is recommended based on the net space saving of 40%.

Also read about the restrictions regarding memory, replication and other fun facts here:

https://msdn.microsoft.com/en-us/library/cc280604(v=sql.110).aspx

As always, your mileage may vary. Test, Test, Test.

CREATE PROCEDURE usp\_identify\_sparse\_candidates

@SchemaName SYSNAME

,@TableName SYSNAME

AS

-- author: Jeff Reinhard

DECLARE @TableRows BIGINT

DECLARE @CommandRows INT

DECLARE @ColumnCounter INT = 1

DECLARE @SQLCommandTable TABLE (SQLLine INT IDENTITY (1,1), SQLCommand VARCHAR(MAX))

DECLARE @SQLString VARCHAR(MAX)

DECLARE @ResultsTable TABLE

(SchemaName SYSNAME,

TableName SYSNAME,

ColumnName SYSNAME,

NullCount INT,

PercentofRows MONEY,

SparseThresholdPct INT,

SparseRecommendation CHAR(20))

SELECT @TableRows = SUM(p.rows)

FROM sys.partitions AS p

INNER JOIN sys.tables AS t ON p.[object\_id] = t.[object\_id]

INNER JOIN sys.schemas AS s ON t.[schema\_id] = s.[schema\_id]

WHERE p.index\_id IN (0,1)

AND t.name = @TableName

AND s.name = @SchemaName

GROUP BY s.NAME, t.NAME

INSERT INTO @SQLCommandTable (SQLCommand)

SELECT 'SELECT '''+@SchemaName+''' as SchemaName, '''+@TableName+''' as TableName, '''+COLUMN\_NAME+''' as ColumnName

,COUNT(1) [NullCount]

,COUNT(1)/'+CAST(@TableRows AS VARCHAR)+'.0\*100 as PercentofRows

,CASE '''+DATA\_TYPE+'''

WHEN ''bit'' THEN 98

WHEN ''tinyint'' THEN 86

WHEN ''smallint'' THEN 76

WHEN ''int'' THEN 64

WHEN ''bigint'' THEN 52

WHEN ''real'' THEN 64

WHEN ''float'' THEN 52

WHEN ''smallmoney'' THEN 64

WHEN ''money'' THEN 52

WHEN ''smalldatetime'' THEN 64

WHEN ''datetime'' THEN 52

WHEN ''uniqueidentifier'' THEN 43

WHEN ''date'' THEN 69

WHEN ''datetime2'' THEN 52

WHEN ''time'' THEN 60

WHEN ''datetimetoffset'' THEN 49

WHEN ''decimal'' THEN 42

WHEN ''numeric'' THEN 42

WHEN ''vardecimal'' THEN 42

WHEN ''varchar'' THEN 60

WHEN ''char'' THEN 60

WHEN ''nvarchar'' THEN 60

WHEN ''nchar'' THEN 60

WHEN ''varbinary'' THEN 60

WHEN ''binary'' THEN 60

WHEN ''xml'' THEN 60

WHEN ''hierarchyid'' THEN 60

ELSE ''UNKNOWN''

END as SparseThresholdPct

,NULL as SparseRecommendation

FROM '+@SchemaName+'.'+@TableName+'

WHERE '+COLUMN\_NAME+' IS NULL;

'

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_SCHEMA = @SchemaName AND TABLE\_NAME = @TableName

AND IS\_NULLABLE = 'YES'

AND DATA\_TYPE NOT IN ('geography','text','geometry','timestamp','image','ntext')

SELECT @CommandRows = COUNT(1) FROM @SQLCommandTable

WHILE @ColumnCounter <= @CommandRows

BEGIN

SELECT @SQLString = SQLCommand FROM @SQLCommandTable WHERE SQLLine = @ColumnCounter

--SELECT @SQLString

INSERT INTO @ResultsTable (SchemaName, TableName, ColumnName, NullCount, PercentofRows, SparseThresholdPct, SparseRecommendation )

EXECUTE (@SQLString)

SET @ColumnCounter=@ColumnCounter+1

END

UPDATE @ResultsTable

SET SparseRecommendation = CASE WHEN PercentofRows >= SparseThresholdPct THEN 'Yes to Sparse' ELSE 'No' END

SELECT \* FROM @ResultsTable

ORDER BY ColumnName

GO