Calculating space used by table and per column

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:27 AM PST

Contents

- Calculate the size of each table (rowcount and storage)
- Calculate the size of each table and index (rowcount and st...
- Calculate the size of each table and index (with index usag...
- Calculate the size of the column data stored in a table
- Public Doc Reference

Scripts to calculate the space used per table, index, and column

This is a "How To" article related to space and storage management.

Calculate the size of each table (rowcount and storage)

```
SELECT
    o.name AS table name,
    s.name AS schema_name,
    p.rows AS row_count,
    SUM(a.total_pages) * 8 AS allocated_pages_KB,
    SUM(a.used_pages) * 8 AS used_pages_KB,
    SUM(a.data_pages) * 8 AS data_pages_KB,
    (SUM(a.total_pages) - SUM(a.used_pages)) * 8 AS unused_pages_KB
    sys.all_objects o
    INNER JOIN sys.indexes i ON o.OBJECT ID = i.object id
    INNER JOIN sys.partitions p ON i.object id = p.OBJECT ID AND i.index id = p.index id
    INNER JOIN sys.allocation units a ON p.partition id = a.container id
    LEFT OUTER JOIN sys.schemas s ON o.schema id = s.schema id
    (o.type = 'U' or o.type like 'V%') and o.is_ms_shipped <> 1
GROUP BY
    o.Name, s.Name, p.Rows
ORDER BY
    o.Name;
```

Sample output:

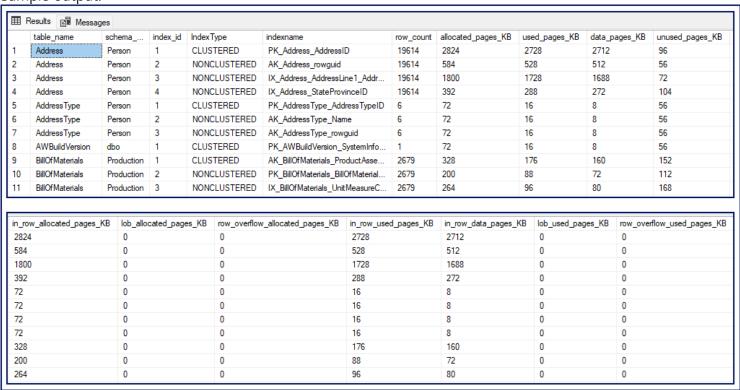
Ⅲ Results ☐ Messages							
	table_name	schema_name	row_count	allocated_pages_KB	used_pages_KB	data_pages_KB	unused_pages_KB
1	Address	Person	19614	5600	5272	5184	328
2	AddressType	Person	6	216	48	24	168
3	AWBuildVersion	dbo	1	72	16	8	56
4	BillOfMaterials	Production	2679	792	360	312	432
5	BusinessEntity	Person	20777	1488	1336	1304	152
6	BusinessEntityAddress	Person	19614	2656	2312	2248	344
7	BusinessEntityContact	Person	909	800	192	128	608
8	ContactType	Person	20	144	32	16	112

Calculate the size of each table and index (rowcount and storage)

The important details can be retrieved from either sys.dm_db_partition_stats or sys.allocation_units:

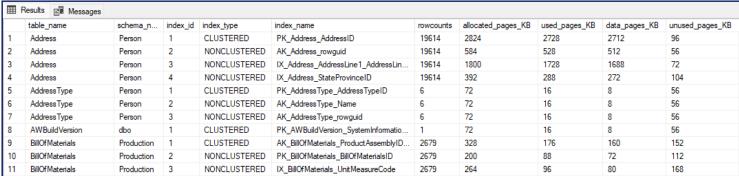
```
SELECT
    o.name AS table_name,
    s.name AS schema name,
    i.index id,
    i.type desc AS IndexType,
    i.name AS indexname,
    ps.row count,
    ps.reserved page count * 8 AS allocated pages KB,
    ps.used page count * 8 AS used pages KB,
    ps.in row data page count * 8 AS data pages KB
    (ps.reserved_page_count - ps.used_page_count) * 8 AS unused_pages_KB,
    ps.in_row_reserved_page_count * 8 AS in_row_allocated_pages_KB,
    ps.lob_reserved_page_count * 8 AS lob_allocated_pages_KB,
    ps.row_overflow_reserved_page_count * 8 AS row_overflow_allocated_pages_KB,
    ps.in_row_used_page_count * 8 AS in_row_used_pages_KB,
    ps.in_row_data_page_count * 8 AS in_row_data_pages_KB,
    ps.lob_used_page_count * 8 AS lob_used_pages_KB,
    ps.row_overflow_used_page_count * 8 AS row_overflow_used_pages_KB
FROM
    sys.all objects o
    INNER JOIN sys.indexes i ON o.object id = i.object id
    INNER JOIN sys.dm db partition stats ps ON ps.[object id] = i.[object id] AND ps.index id = i.index id
    LEFT OUTER JOIN sys.schemas s ON o.schema_id = s.schema_id
    (o.type = 'U' or o.type like 'V%') and o.is_ms_shipped <> 1
ORDER BY
    o.name, i.index id;
```

Sample output:



```
-- similar output with fewer details, retrieved from a different source to cross-check:
    o.name AS table name,
    s.Name AS schema_name,
    i.index id,
    i.type desc AS index type,
    i.name AS index name,
    p.rows AS rowcounts,
    SUM(a.total pages) * 8 AS allocated pages KB,
    SUM(a.used_pages) * 8 AS used pages KB,
    SUM(a.data pages) * 8 AS data pages KB,
    (SUM(a.total pages) - SUM(a.used pages)) * 8 AS unused pages KB
    sys.all objects o
    INNER JOIN sys.indexes i ON o.OBJECT ID = i.object id
    INNER JOIN sys.partitions p ON i.object id = p.OBJECT ID AND i.index id = p.index id
    INNER JOIN sys.allocation units a ON p.partition id = a.container id
    LEFT OUTER JOIN sys.schemas s ON o.schema_id = s.schema_id
    (o.type = 'U' or o.type like 'V%') and o.is_ms_shipped <> 1
GROUP BY
   o.Name, s.Name, i.type desc, i.name, i.index id, p.Rows
ORDER BY o.Name, i.index id;
```

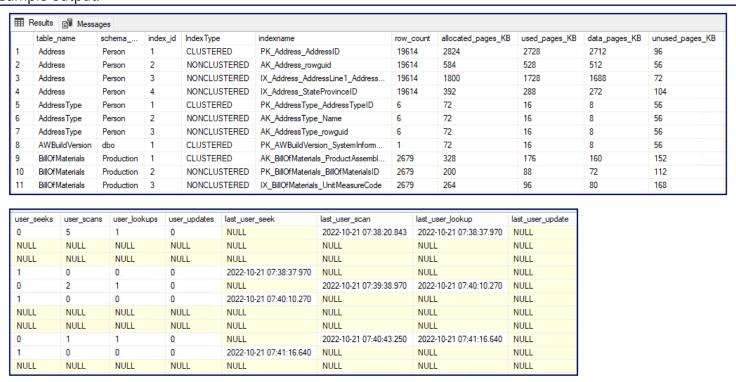
Sample output:



Calculate the size of each table and index (with index usage statistics where available)

```
SELECT
    o.name AS table name,
    s.name AS schema_name,
    i.index id,
    i.type desc AS IndexType,
    i.name AS indexname,
    ps.row count,
    ps.reserved page count * 8 AS allocated pages KB,
    ps.used page count * 8 AS used pages KB,
    ps.in_row_data_page_count * 8 AS data pages KB,
    (ps.reserved_page_count - ps.used_page_count) * 8 AS unused pages KB,
    ius.user_seeks, ius.user_scans, ius.user_lookups, ius.user updates,
    ius.last user seek, ius.last user scan, ius.last user lookup, ius.last user update
FROM
    sys.all objects o
    INNER JOIN sys.indexes i ON o.object id = i.object id
    INNER JOIN sys.dm db partition stats ps ON ps.[object id] = i.object id AND ps.index id = i.index id
    LEFT JOIN sys.dm db index usage stats ius ON ius.object id = i.object id AND ius.index id = i.index id
    LEFT JOIN sys.schemas s ON o.schema id = s.schema id
WHERE
    (o.type = 'U' or o.type like 'V%') and o.is ms shipped <> 1
ORDER BY
    o.name, i.index id;
```

Sample output:



Calculate the size of the column data stored in a table

This query runs on a per-table basis, because the calculation is very expensive. Set the requested table name as "schema.table" e.g. "dbo.table1" into the @ObjectTable variable.

```
DECLARE @ObjectTable
                         sysname = 'Person.Person'
DECLARE @Name
                         sysname
DECLARE @Type
                         sysname
DECLARE @Length
                         INT
DECLARE @Total
                         BIGINT = 0
DECLARE @SQL
                         NVARCHAR (4000)
DECLARE @ParamDefinition NVARCHAR(100) = N'@Total BIGINT OUTPUT'
SET ANSI WARNINGS OFF
DECLARE vColumns CURSOR FOR
    SELECT c.name, t.name as 'type', c.max length
    FROM sys.columns c
    INNER JOIN sys.types t ON c.system_type_id = t.user_type_id
    WHERE object id = OBJECT ID(@ObjectTable)
    ORDER BY c.column id
OPEN vColumns
FETCH NEXT FROM vColumns INTO @Name, @Type, @Length
WHILE @@FETCH STATUS = 0
BEGIN
    SET @Total = 0
    SET @SQL = 'SELECT @Total = SUM(DATALENGTH(' + @Name + ')) FROM ' + @ObjectTable
    EXECUTE sp_executesql @SQL, @ParamDefinition, @Total = @Total OUTPUT;
    PRINT + 'Size (KB): ' + CONVERT(CHAR(20), (@Total / 1024)) + 'Column: '+ CONVERT(VARCHAR(128), @Name) + '
    FETCH NEXT FROM vColumns INTO @Name, @Type, @Length
END
CLOSE vColumns;
DEALLOCATE vColumns;
/* sample output:
Size (KB): 78
                               Column: BusinessEntityID (type: int, length: 4)
Size (KB): 78
                               Column: PersonType (type: nchar, length: 4)
Size (KB): 19
                               Column: NameStyle (type: bit, length: 1)
Size (KB): 5
                               Column: Title (type: nvarchar, length: 16)
Size (KB): 230
                               Column: FirstName (type: nvarchar, length: 100)
Size (KB): 23
                               Column: MiddleName (type: nvarchar, length: 100)
Size (KB): 218
                               Column: LastName (type: nvarchar, length: 100)
Size (KB): 0
                               Column: Suffix (type: nvarchar, length: 20)
Size (KB): 78
                               Column: EmailPromotion (type: int, length: 4)
Size (KB): 19
                               Column: AdditionalContactInfo (type: xml, length: -1)
Size (KB): 24073
                               Column: Demographics (type: xml, length: -1)
Size (KB): 312
                               Column: rowguid (type: uniqueidentifier, length: 16)
Size (KB): 156
                               Column: ModifiedDate (type: datetime, length: 8)
```

Public Doc Reference

• Blog post: Lesson Learned 150: Calculating the space used by table and per column [2]

How good have you found this content?

