Efficiently convert rows to columns in sql server

stackoverflow.com/questions/15745042/efficiently-convert-rows-to-columns-in-sql-server

399

I'm looking for an efficient way to convert rows to columns in SQL server, I heard that PIVOT is not very fast, and I need to deal with lot of records.

This is my example:

```
------------------------------
| Id | Value  | ColumnName    |
------------------------------
| 1  | John   | FirstName     |
| 2  | 2.4    | Amount        |
| 3  | ZH1E4A | PostalCode    |
| 4  | Fork   | LastName      |
| 5  | 857685 | AccountNumber |
------------------------------
```

This is my result:

```
-----------------------------------------------------------------------
| FirstName  |Amount|   PostalCode   |   LastName  |  AccountNumber |
-----------------------------------------------------------------------
| John       | 2.4  |   ZH1E4A       |   Fork      |  857685        |
-----------------------------------------------------------------------
```

How can I build the result?

sql sql-server sql-server-2008 pivot
edited Sep 14 '18 at 21:21
A Friend
asked Apr 1 '13 at 14:11

tbag
add a comment

2 Answers

Active Oldest Votes
570

There are several ways that you can transform data from multiple rows into columns.

Using `PIVOT`

In SQL Server you can use the `PIVOT` function to transform the data from rows to columns:

```
select Firstname, Amount, PostalCode, LastName, AccountNumber
from
(
  select value, columnname
  from yourtable
) d
pivot
(
  max(value)
  for columnname in (Firstname, Amount, PostalCode, LastName, AccountNumber)
) piv;
```

See Demo.

Pivot with unknown number of `columnnames`

If you have an unknown number of `columnnames` that you want to transpose, then you can use dynamic SQL:

```
DECLARE @cols AS NVARCHAR(MAX),
    @query  AS NVARCHAR(MAX)

select @cols = STUFF((SELECT ',' + QUOTENAME(ColumnName)
                      from yourtable
                      group by ColumnName, id
                      order by id
            FOR XML PATH(''), TYPE
            ).value('.', 'NVARCHAR(MAX)')
        ,1,1,'')

set @query = N'SELECT ' + @cols + N' from
             (
                select value, ColumnName
                from yourtable
            ) x
            pivot
            (
                max(value)
                for ColumnName in (' + @cols + N')
            ) p '

exec sp_executesql @query;
```

See Demo.

Using an aggregate function

If you do not want to use the `PIVOT` function, then you can use an aggregate function with a `CASE` expression:

```
select
  max(case when columnname = 'FirstName' then value end) Firstname,
  max(case when columnname = 'Amount' then value end) Amount,
  max(case when columnname = 'PostalCode' then value end) PostalCode,
  max(case when columnname = 'LastName' then value end) LastName,
  max(case when columnname = 'AccountNumber' then value end) AccountNumber
from yourtable
```

See Demo.

Using multiple joins

This could also be completed using multiple joins, but you will need some column to associate each of the rows which you do not have in your sample data. But the basic syntax would be:

```
select fn.value as FirstName,
  a.value as Amount,
  pc.value as PostalCode,
  ln.value as LastName,
  an.value as AccountNumber
from yourtable fn
left join yourtable a
  on fn.somecol = a.somecol
  and a.columnname = 'Amount'
left join yourtable pc
  on fn.somecol = pc.somecol
  and pc.columnname = 'PostalCode'
left join yourtable ln
  on fn.somecol = ln.somecol
  and ln.columnname = 'LastName'
left join yourtable an
  on fn.somecol = an.somecol
  and an.columnname = 'AccountNumber'
where fn.columnname = 'Firstname'
```

edited Sep 17 '19 at 18:23
jpaugh
answered Apr 1 '13 at 14:13

Taryn♦

- 6
- 1
- 11

  @tbag If you have an unknown number of rows, then you would have to use dynamic sql but be aware that transforming millions of rows will not be efficient. – Taryn♦ Apr 1 '13 at 14:33
- 1

show **18** more comments

11

This is rather a method than just a single script but gives you much more flexibility.

First of all There are 3 objects:

1. User defined TABLE type [`ColumnActionList`] -> holds data as parameter
2. SP [`proc_PivotPrepare`] -> prepares our data
3. SP [`proc_PivotExecute`] -> execute the script

CREATE TYPE [dbo].[ColumnActionList] AS TABLE ([ID] [smallint] NOT NULL, [ColumnName] nvarchar NOT NULL, [Action] nchar NOT NULL); GO

```
    CREATE PROCEDURE [dbo].[proc_PivotPrepare]
    (
    @DB_Name        nvarchar(128),
    @TableName      nvarchar(128)
    )
    AS
            SELECT @DB_Name = ISNULL(@DB_Name,db_name())
    DECLARE @SQL_Code nvarchar(max)

    DECLARE @MyTab TABLE (ID smallint identity(1,1), [Column_Name] nvarchar(128),
[Type] nchar(1), [Set Action SQL] nvarchar(max));

    SELECT @SQL_Code        =   'SELECT [<| SQL_Code |>] = '' '' '
                                    + 'UNION ALL '
                                    + 'SELECT ''------------------------------
-----------------------------------------------------------------'' '
                                    + 'UNION ALL '
                                    + 'SELECT ''-----| Declare user defined
type [ID] / [ColumnName] / [PivotAction] '' '
                                    + 'UNION ALL '
                                    + 'SELECT ''------------------------------
-----------------------------------------------------------------'' '
                                    + 'UNION ALL '
                                    + 'SELECT ''DECLARE @ColumnListWithActions
ColumnActionList;'''
                                    + 'UNION ALL '
                                    + 'SELECT ''------------------------------
-----------------------------------------------------------------'' '
                                    + 'UNION ALL '
                                    + 'SELECT ''-----| Set [PivotAction]
(''''S'''' as default) to select dimentions and values '' '
                                    + 'UNION ALL '
                                    + 'SELECT ''-----|'''
                                    + 'UNION ALL '
                                    + 'SELECT ''-----| ''''S'''' = Stable
column || ''''D'''' = Dimention column || ''''V'''' = Value column '' '
                                    + 'UNION ALL '
                                    + 'SELECT ''------------------------------
-----------------------------------------------------------------'' '
                                    + 'UNION ALL '
                                    + 'SELECT ''INSERT INTO
@ColumnListWithActions VALUES ('' + CAST( ROW_NUMBER() OVER (ORDER BY [NAME]) as
nvarchar(10)) + '', '' + ''''''''  + [NAME] + ''''''''+ '', ''''S'''');'''
                                    + 'FROM [' + @DB_Name + '].sys.columns  '
                                    + 'WHERE object_id = object_id(''[' +
@DB_Name + ']..[' + @TableName + ']'') '
                                    + 'UNION ALL '
                                    + 'SELECT ''------------------------------
-----------------------------------------------------------------'' '
                                    + 'UNION ALL '
                                    + 'SELECT ''-----| Execute sp_PivotExecute
with parameters: columns and dimentions and main table name'' '
                                    + 'UNION ALL '
                                    + 'SELECT ''------------------------------
-----------------------------------------------------------------'' '
                                    + 'UNION ALL '
                                    + 'SELECT ''EXEC [dbo].[sp_PivotExecute]
@ColumnListWithActions, ' + '''''' + @TableName + '''''' + ';'''
                                    + 'UNION ALL '
```

```
                                        + 'SELECT ''----------------------------
------------------------------------------------------------------''' '
EXECUTE SP_EXECUTESQL @SQL_Code;

GO

CREATE PROCEDURE [dbo].[sp_PivotExecute]
(
@ColumnListWithActions  ColumnActionList ReadOnly
,@TableName                     nvarchar(128)
)
AS


--
#############################################################################################

--###| Step 1 - Select our user-defined-table-variable into temp table
--
#############################################################################################


IF OBJECT_ID('tempdb.dbo.#ColumnListWithActions', 'U') IS NOT NULL DROP TABLE
#ColumnListWithActions;
SELECT * INTO #ColumnListWithActions FROM @ColumnListWithActions;

--
#############################################################################################

--###| Step 2 - Preparing lists of column groups as strings:
--
#############################################################################################


DECLARE @ColumnName                     nvarchar(128)
DECLARE @Destiny                        nchar(1)

DECLARE @ListOfColumns_Stable           nvarchar(max)
DECLARE @ListOfColumns_Dimension    nvarchar(max)
DECLARE @ListOfColumns_Variable     nvarchar(max)
--##############################
--###| Cursor for List of Stable Columns
--##############################

DECLARE ColumnListStringCreator_S CURSOR FOR
SELECT      [ColumnName]
FROM        #ColumnListWithActions
WHERE       [Action] = 'S'
OPEN ColumnListStringCreator_S;
FETCH NEXT FROM ColumnListStringCreator_S
INTO @ColumnName
  WHILE @@FETCH_STATUS = 0

   BEGIN
        SELECT @ListOfColumns_Stable = ISNULL(@ListOfColumns_Stable, '') + ' [' +
@ColumnName + '] ,';
        FETCH NEXT FROM ColumnListStringCreator_S INTO @ColumnName
   END
```

```
CLOSE ColumnListStringCreator_S;
DEALLOCATE ColumnListStringCreator_S;

--############################
--###| Cursor for List of Dimension Columns
--############################

DECLARE ColumnListStringCreator_D CURSOR FOR
SELECT      [ColumnName]
FROM        #ColumnListWithActions
WHERE       [Action] = 'D'
OPEN ColumnListStringCreator_D;
FETCH NEXT FROM ColumnListStringCreator_D
INTO @ColumnName
  WHILE @@FETCH_STATUS = 0

   BEGIN
        SELECT @ListOfColumns_Dimension = ISNULL(@ListOfColumns_Dimension, '') + '
[' + @ColumnName + '] ,';
        FETCH NEXT FROM ColumnListStringCreator_D INTO @ColumnName
   END

CLOSE ColumnListStringCreator_D;
DEALLOCATE ColumnListStringCreator_D;

--############################
--###| Cursor for List of Variable Columns
--############################

DECLARE ColumnListStringCreator_V CURSOR FOR
SELECT      [ColumnName]
FROM        #ColumnListWithActions
WHERE       [Action] = 'V'
OPEN ColumnListStringCreator_V;
FETCH NEXT FROM ColumnListStringCreator_V
INTO @ColumnName
  WHILE @@FETCH_STATUS = 0

   BEGIN
        SELECT @ListOfColumns_Variable = ISNULL(@ListOfColumns_Variable, '') + '
[' + @ColumnName + '] ,';
        FETCH NEXT FROM ColumnListStringCreator_V INTO @ColumnName
   END

CLOSE ColumnListStringCreator_V;
DEALLOCATE ColumnListStringCreator_V;

SELECT @ListOfColumns_Variable      = LEFT(@ListOfColumns_Variable,
LEN(@ListOfColumns_Variable) - 1);
SELECT @ListOfColumns_Dimension = LEFT(@ListOfColumns_Dimension,
LEN(@ListOfColumns_Dimension) - 1);
SELECT @ListOfColumns_Stable            = LEFT(@ListOfColumns_Stable,
LEN(@ListOfColumns_Stable) - 1);

--
################################################################################

--###| Step 3 - Preparing table with all possible connections between Dimension
columns excluding NULLs
```

```
--
#############################################################################################

DECLARE @DIM_TAB TABLE ([DIM_ID] smallint, [ColumnName] nvarchar(128))
INSERT INTO @DIM_TAB
SELECT [DIM_ID] = ROW_NUMBER() OVER(ORDER BY [ColumnName]), [ColumnName] FROM
#ColumnListWithActions WHERE [Action] = 'D';

DECLARE @DIM_ID smallint;
SELECT      @DIM_ID = 1;


DECLARE @SQL_Dimentions nvarchar(max);

IF OBJECT_ID('tempdb.dbo.##ALL_Dimentions', 'U') IS NOT NULL DROP TABLE
##ALL_Dimentions;

SELECT @SQL_Dimentions      = 'SELECT [xxx_ID_xxx] = ROW_NUMBER() OVER (ORDER BY '
+ @ListOfColumns_Dimension + '), ' + @ListOfColumns_Dimension
                                        + ' INTO ##ALL_Dimentions '
                                        + ' FROM (SELECT DISTINCT' +
@ListOfColumns_Dimension + ' FROM  ' + @TableName
                                        + ' WHERE ' + (SELECT [ColumnName]
FROM @DIM_TAB WHERE [DIM_ID] = @DIM_ID) + ' IS NOT NULL ';
                                        SELECT @DIM_ID = @DIM_ID + 1;
            WHILE @DIM_ID <= (SELECT MAX([DIM_ID]) FROM @DIM_TAB)
            BEGIN
            SELECT @SQL_Dimentions = @SQL_Dimentions + 'AND ' + (SELECT
[ColumnName] FROM @DIM_TAB WHERE [DIM_ID] = @DIM_ID) +  ' IS NOT NULL ';
            SELECT @DIM_ID = @DIM_ID + 1;
            END

SELECT @SQL_Dimentions   = @SQL_Dimentions + ' )x';

EXECUTE SP_EXECUTESQL  @SQL_Dimentions;

--
#############################################################################################

--###| Step 4 - Preparing table with all possible connections between Stable
columns excluding NULLs
--
#############################################################################################

DECLARE @StabPos_TAB TABLE ([StabPos_ID] smallint, [ColumnName] nvarchar(128))
INSERT INTO @StabPos_TAB
SELECT [StabPos_ID] = ROW_NUMBER() OVER(ORDER BY [ColumnName]), [ColumnName] FROM
#ColumnListWithActions WHERE [Action] = 'S';

DECLARE @StabPos_ID smallint;
SELECT      @StabPos_ID = 1;


DECLARE @SQL_MainStableColumnTable nvarchar(max);

IF OBJECT_ID('tempdb.dbo.##ALL_StableColumns', 'U') IS NOT NULL DROP TABLE
##ALL_StableColumns;

SELECT @SQL_MainStableColumnTable       = 'SELECT xxx_ID_xxx = ROW_NUMBER() OVER
```

```
(ORDER BY ' + @ListOfColumns_Stable + '), ' + @ListOfColumns_Stable
                                            + ' INTO ##ALL_StableColumns '
                                            + ' FROM (SELECT DISTINCT' +
@ListOfColumns_Stable + ' FROM  ' + @TableName
                                            + ' WHERE ' + (SELECT [ColumnName]
FROM @StabPos_TAB WHERE [StabPos_ID] = @StabPos_ID) + ' IS NOT NULL ';
                                            SELECT @StabPos_ID = @StabPos_ID + 1;
            WHILE @StabPos_ID <= (SELECT MAX([StabPos_ID]) FROM @StabPos_TAB)
            BEGIN
            SELECT @SQL_MainStableColumnTable = @SQL_MainStableColumnTable + 'AND
' + (SELECT [ColumnName] FROM @StabPos_TAB WHERE [StabPos_ID] = @StabPos_ID) +  '
IS NOT NULL ';
            SELECT @StabPos_ID = @StabPos_ID + 1;
            END

SELECT @SQL_MainStableColumnTable    = @SQL_MainStableColumnTable + ' )x';

EXECUTE SP_EXECUTESQL  @SQL_MainStableColumnTable;

--
#############################################################################################

--###| Step 5 - Preparing table with all options ID
--
#############################################################################################


DECLARE @FULL_SQL_1 NVARCHAR(MAX)
SELECT @FULL_SQL_1 = ''

DECLARE @i smallint

IF OBJECT_ID('tempdb.dbo.##FinalTab', 'U') IS NOT NULL DROP TABLE ##FinalTab;

SELECT @FULL_SQL_1 = 'SELECT t.*, dim.[xxx_ID_xxx] '
                                    + ' INTO ##FinalTab '
                                    +   'FROM ' + @TableName + ' t '
                                    +   'JOIN ##ALL_Dimentions dim '
                                    +   'ON t.' + (SELECT [ColumnName] FROM
@DIM_TAB WHERE [DIM_ID] = 1) + ' = dim.' + (SELECT [ColumnName] FROM @DIM_TAB
WHERE [DIM_ID] = 1);
                                SELECT @i = 2
                                WHILE @i <= (SELECT MAX([DIM_ID]) FROM @DIM_TAB)
                                    BEGIN
                                    SELECT @FULL_SQL_1 = @FULL_SQL_1 + ' AND t.' +
(SELECT [ColumnName] FROM @DIM_TAB WHERE [DIM_ID] = @i) + ' = dim.' + (SELECT
[ColumnName] FROM @DIM_TAB WHERE [DIM_ID] = @i)
                                    SELECT @i = @i +1
                                END
EXECUTE SP_EXECUTESQL @FULL_SQL_1

--
#############################################################################################

--###| Step 6 - Selecting final data
--
#############################################################################################

DECLARE @STAB_TAB TABLE ([STAB_ID] smallint, [ColumnName] nvarchar(128))
```

```
INSERT INTO @STAB_TAB
SELECT [STAB_ID] = ROW_NUMBER() OVER(ORDER BY [ColumnName]), [ColumnName]
FROM #ColumnListWithActions WHERE [Action] = 'S';

DECLARE @VAR_TAB TABLE ([VAR_ID] smallint, [ColumnName] nvarchar(128))
INSERT INTO @VAR_TAB
SELECT [VAR_ID] = ROW_NUMBER() OVER(ORDER BY [ColumnName]), [ColumnName]
FROM #ColumnListWithActions WHERE [Action] = 'V';

DECLARE @y smallint;
DECLARE @x smallint;
DECLARE @z smallint;


DECLARE @FinalCode nvarchar(max)

SELECT @FinalCode = ' SELECT ID1.*'
                                        SELECT @y = 1
                                        WHILE @y <= (SELECT MAX([xxx_ID_xxx]) FROM
##FinalTab)
                                            BEGIN
                                                SELECT @z = 1
                                                WHILE @z <= (SELECT MAX([VAR_ID])
FROM @VAR_TAB)
                                                    BEGIN
                                                        SELECT @FinalCode =
@FinalCode +    ', [ID' + CAST((@y) as varchar(10)) + '.' + (SELECT [ColumnName]
FROM @VAR_TAB WHERE [VAR_ID] = @z) + '] =  ID' + CAST((@y + 1) as varchar(10)) +
'.' + (SELECT [ColumnName] FROM @VAR_TAB WHERE [VAR_ID] = @z)
                                                        SELECT @z = @z + 1
                                                    END
                                                    SELECT @y = @y + 1
                                                END
        SELECT @FinalCode = @FinalCode +
                                        ' FROM ( SELECT * FROM
##ALL_StableColumns)ID1';
                                        SELECT @y = 1
                                        WHILE @y <= (SELECT MAX([xxx_ID_xxx]) FROM
##FinalTab)
                                        BEGIN
                                            SELECT @x = 1
                                            SELECT @FinalCode = @FinalCode
                                                                                +
' LEFT JOIN (SELECT ' +  @ListOfColumns_Stable + ' , ' + @ListOfColumns_Variable
                                                                                +
' FROM ##FinalTab WHERE [xxx_ID_xxx] = '
                                                                                +
CAST(@y as varchar(10)) + ' )ID' + CAST((@y + 1) as varchar(10))
                                                                                +
' ON 1 = 1'

WHILE @x <= (SELECT MAX([STAB_ID]) FROM @STAB_TAB)

BEGIN

SELECT @FinalCode = @FinalCode + ' AND ID1.' + (SELECT [ColumnName] FROM @STAB_TAB
WHERE [STAB_ID] = @x) + ' = ID' + CAST((@y+1) as varchar(10)) + '.' + (SELECT
[ColumnName] FROM @STAB_TAB WHERE [STAB_ID] = @x)
```

```
SELECT @x = @x +1

END
                                                SELECT @y = @y + 1
                                            END

SELECT * FROM ##ALL_Dimentions;
EXECUTE SP_EXECUTESQL @FinalCode;
```

From executing the first query (by passing source DB and table name) you will get a pre-created execution query for the second SP, all you have to do is define is the column from your source: + Stable + Value (will be used to concentrate values based on that) + Dim (column you want to use to pivot by)

Names and datatypes will be defined automatically!

I cant recommend it for any production environments but does the job for adhoc BI requests.

edited Jun 9 at 16:28

answered Jan 23 '17 at 16:01

Bartosz X

- Thanks Bartosz, managed to use some of the ideas from your script and done what I had on my mind already, but nevertheless, thanks for updating it :) . I should have thought to change that line, but honestly thought is a stored procedure you've forgot is not default in the system or something like that. I will give it a run when i get close to that project again, and update here! – FAB Sep 25 '18 at 18:35
- 1

add a comment

🔥

Highly active question. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.

Not the answer you're looking for? Browse other questions tagged sql sql-server sql-server-2008 pivot or ask your own question.