How can I use an SQL Pivot for this?

Asked 16 years, 1 month ago Modified 12 years, 4 months ago Viewed 1k times

1

I have a data set that is organized in the following manner:

```
Timestamp|A0001|A0002|A0003|A0004|B0001|B0002|B0003|B0004 ...
---------+-----+-----+-----+-----+-----+-----+-----+-----
2008-1-1 |  1  |  2  | 10  |   6 |  20 |  35 | 300 |  8
2008-1-2 |  5  |  2  |  9  |   3 |  50 |  38 | 290 |  2
2008-1-4 |  7  |  7  | 11  |   0 |  30 |  87 | 350 |  0
2008-1-5 |  1  |  9  |  1  |   0 |  25 | 100 |  10 |  0
...
```

Where A0001 is Value A of item #1 and B0001 is Value B of item #1. There can be over 60 different items in a table, and each item has an A value column and a B value column, meaning a total of over 120 columns in the table.

Where I want to get to is a 3 column result (Item index, A Value, B Value) that sums the A and B values for each item:

```
Index | A Value | B Value
------+---------+--------
 0001 |   14    |   125
 0002 |   20    |   260
 0003 |   31    |   950
 0004 |    9    |    10
 ....
```

As I am going from columns to rows I would expect a pivot in the solution, but I am not sure of how to flesh it out. Part of the issue is how to strip out the A's and B's to form the values for the Index column. The other part is that I have never had to use a Pivot before, so I am stumbling over the basic syntax as well.

I think that ultimately I need to have a multi step solution that first builds the summations as:

```
ColName | Value
--------+------
A0001   |  14
A0002   |  20
A0003   |  31
A0004   |   9
B0001   | 125
B0002   | 260
B0003   | 950
B0004   |  10
```

Then modify the ColName data to strip out the index:

```
ColName | Value | Index | Aspect
--------+-------+-------+-------
A0001   |  14   | 0001  |  A
A0002   |  20   | 0002  |  A
A0003   |  31   | 0003  |  A
A0004   |   9   | 0004  |  A
B0001   | 125   | 0001  |  B
B0002   | 260   | 0002  |  B
B0003   | 950   | 0003  |  B
B0004   |  10   | 0004  |  B
```

Finally self join to move the B values up next to the A Values.

This seems to be a long winded process to get what I want. So I am after advice as to whether I am headed down the right path, or is there another approach that I have over looked that will make my life so much easier.

Note 1) The solution has to be in T-SQL on MSSQL 2005.

Note 2) The format of the table cannot be changed.

Edit Another method I have thought about uses UNIONs and individual SUM()s on each column:

```
SELECT '0001' as Index, SUM(A0001) as A, SUM(B0001) as B FROM TABLE
UNION
SELECT '0002' as Index, SUM(A0002) as A, SUM(B0002) as B FROM TABLE
UNION
SELECT '0003' as Index, SUM(A0003) as A, SUM(B0003) as B FROM TABLE
UNION
SELECT '0004' as Index, SUM(A0004) as A, SUM(B0004) as B FROM TABLE
UNION
...
```

But this approach really doesn't look very nice either

EDIT So far there are 2 great responses. But I would like to add two more conditions to the query :-)

1) I need to select the rows based on a range of timestamps (minv < timestamp < maxv).

2) I also need to conditionally select rows on a UDF that processes the timestamp

Using Brettski's table names, would the above translate to:

```
...
(SELECT A0001, A0002, A0003, B0001, B0002, B0003
```

```
 FROM ptest
 WHERE timestamp>minv AND timestamp<maxv AND fn(timestamp)=fnv) p
unpivot
(val for item in (A0001, A0002, A0003, B0001, B0002, B0003)) as unpvt
...
```

Given that I have conditionally add the fn() requirement, I think that I also need to go down the dynamic SQL path as proposed by Jonathon. Especially as I have to build the same query for 12 different tables - all of the same style.

sql-server sql-server-2005 t-sql pivot

Share
Improve this question
Follow

edited Aug 6, 2012 at 19:58
user212218

asked Nov 21, 2008 at 14:20
Peter M
7,483 ● 3 ● 53 ● 94

2 Answers

Sorted by: Highest score (default)

5

Same kinda answer here, that was fun:

```
-- Get column names from system table
DECLARE @phCols NVARCHAR(2000)
SELECT @phCols = COALESCE(@phCols + ',[' + name + ']', '[' + name + ']')
    FROM syscolumns WHERE id = (select id from sysobjects where name = 'Test'
and type='U')

-- Get rid of the column we don't want
SELECT @phCols = REPLACE(@phCols, '[Timestamp],', '')

-- Query & sum using the dynamic column names
DECLARE @exec nvarchar(2000)
SELECT @exec =
'
    select
        SUBSTRING([Value], 2, LEN([Value]) - 1) as [Index],
        SUM(CASE WHEN (LEFT([Value], 1) = ''A'') THEN Cols ELSE 0 END) as
AValue,
        SUM(CASE WHEN (LEFT([Value], 1) = ''B'') THEN Cols ELSE 0 END) as
BValue
    FROM
    (
        select *
        from (select ' + @phCols + ' from Test) as t
        unpivot (Cols FOR [Value] in (' + @phCols + ')) as p
    ) _temp
    GROUP BY SUBSTRING([Value], 2, LEN([Value]) - 1)
'
EXECUTE(@exec)
```

You don't need to hard code column names in this one.

Share

Improve this answer

Follow

edited Nov 21, 2008 at 16:47

answered Nov 21, 2008 at 16:31

Jonathan DeMarks

144 ● 1 ● 7

Thanks Jonathan. I was thinking of that way to eliminate the hard coding of columns, but didn't put it together. I hated to think of listing 160+ columns. :) – Brettski Nov 21, 2008 at 17:18

You can add you timestamp where clause to this query as well, just change [[from (select ' + @phCols + ' from Test) as t]] to [[from (select ' + @phCols + ' from Test WHERE timestamp>minv AND timestamp<maxv AND fn(timestamp)=fnv) as t]] – Jonathan DeMarks Nov 21, 2008 at 17:27

Its actually worse than that .. much much worse. 12 sets of tables, and each set has 123 unique item names, for a total of 1476 **items* or 2952 column names. A maintenance nightmare! – Peter M Nov 21, 2008 at 17:30

And of course when I ran the solution over my actual tables I hit the limit of 4000 chars being passed to exec! So I had to split the solution into two parts. – Peter M Nov 21, 2008 at 18:21

1

OK, I have come up with one solution which should get you started. It will probably take some time to put together, but will perform well. It would be nice if we didn't have to list out all the columns by name.

Basically this is using UNPIVOT and placing that product into a temp table, then querying it into your final data set. I named my table ptest when I put this together, this is the one with all of the A0001, etc columns.

```
-- Create the temp table
CREATE TABLE #s (item nvarchar(10), val int)

-- Insert UNPIVOT product into the temp table
INSERT INTO  #s (item, val)
SELECT item, val
FROM
(SELECT A0001, A0002, A0003, B0001, B0002, B0003
FROM ptest) p
unpivot
(val for item in (A0001, A0002, A0003, B0001, B0002, B0003)) as unpvt

-- Query the temp table to get final data set
SELECT RIGHT(item, 4) as item1,
Sum(CASE WHEN LEFT(item, 1) = 'A' THEN val ELSE 0 END) as A,
Sum(CASE WHEN LEFT(item, 1) = 'B' THEN val ELSE 0 END) as B
from #s
GROUP BY RIGHT(item, 4)

-- Delete temp table
drop table #s
```

By the way, thanks for the question, this was the first time I got to use UNPIVOT. Always wanted to, just never had a need.

Share Improve this answer Follow

answered Nov 21, 2008 at 16:12

Brettski

20k ● 17 ● 77 ● 101

Thanks for that. I can sort of see where you are coming from but it is making head hurt. I do have one small ruffle related to the SELECT FROM ptest which I think I should have mentioned. I need to select the initial rows based on a range of timestamps (which is not an issue) – Peter M Nov 21, 2008 at 16:37

But also I need to select based on a function of the timestamp. So would I have: (SELECT fn(timestamp) as fnv, A001 .. FROM ptest WHERE timestamp<XX and fnv=yyy) and then only unpivot on the A's andB's ? – Peter M Nov 21, 2008 at 16:40

Yeah, that complicates things a bit, doesn't it. – Brettski Nov 21, 2008 at 17:42