Paper 9660-2016

Performing Efficient Wide-To-Long Transposes on Teradata Tables Using SAS Explicit Pass-Through

Tao Cheng, Accenture, Almere, the Netherlands

ABSTRACT

SAS® provides in-database processing technology in PROC SQL, which allows the SQL explicit pass-through method to push some or all of the work to a database management system (DBMS). This paper focuses on using the SAS SQL explicit pass-through method to transform Teradata table columns into rows.

There are two common approaches for transforming table columns into rows. The first approach is to create 'narrow' tables, one for each column that requires transposition, and then use UNION or UNION ALL to append all the tables together. This approach is straightforward but can be quite cumbersome, especially when there is a large number of columns that need to be transposed. The second approach is using the Teradata TD_UNPIVOT function which makes the wide-to-long table transposition an easy job. However, TD_UNPIVOT only allows you to transpose columns with same data type from wide to long. This paper presents a SAS macro solution to the wide-to-long table transposition involving different column data types. Several examples are provided to illustrate the usage of the macro solution.

This paper complements the author's SAS paper "Performing Efficient Transposes on Large Teradata Tables Using SQL Explicit Pass-Through" in which the solution of performing long-to-wide table transposition method is discussed. SAS Programmers who are working with data stored in an external DBMS and would like to efficiently transpose their data will benefit from this paper.

INTRODUCTION

Wide-to-long table transposition is a popular data manipulation practice. The transposed data can be used in subsequent DATA or PROC steps for analysis, reporting, or further data manipulation.

The Teradata TD_UNPIVOT function is a simple and powerful function. You can use TD_UNPIVOT to turn variables into observations. An example is given below.

INPUT TABLE

sales_person	sales_q1	sales_q2	sales_q3	sales_q4	cost_q1	cost_q2	cost_q3	cost_q4
Jack	100.12	90.25	102.70	.	78.34	75.20	77.25	.
Lisa	89.12	109.75	112.67	105.89	76.34	77.20	76.75	81.70

Display 1. Input Table Before TD_UNPIVOT transposition

OUTPUT TABLE

sales_person	period	sales	cost
Jack	q1	100.12	78.34
Lisa	q1	89.12	76.34
Jack	q2	90.25	75.20
Lisa	q2	109.75	77.20
Jack	q3	102.70	77.25
Lisa	q3	112.67	76.75
Jack	q4	.	.
Lisa	q4	105.89	81.70

Display 2. Output Table After TD_UNPIVOT transposition

The code using the TD_UNPIVOT function performs the wide-to-long transposition is as follows.

```
PROC SQL;
  CONNECT TO teradata (SERVER=testserver USER=testuser PASSWORD=testpass);
  EXECUTE ( CREATE TABLE testdatabase.tbl_output AS(
              SELECT * FROM TD_UNPIVOT
               (
                ON(SELECT * FROM testdatabase.tbl_input)
                USING
                 VALUE_COLUMNS('sales','cost')
                 UNPIVOT_COLUMN('period')
                 COLUMN_LIST('sales_q1, cost_q1',
                             'sales_q2, cost_q2',
                             'sales_q3, cost_q3',
                             'sales_q4, cost_q4')
                 COLUMN_ALIAS_LIST('q1', 'q2', 'q3', 'q4')
                 INCLUDE_NULLS('Yes')
               )X
            ) WITH DATA PRIMARY INDEX(sales_person,period)
          ) BY teradata;
  EXECUTE (COMMIT) BY teradata;
QUIT;
```

- By default SAS/ACCESS to Teradata uses ANSI Mode and ANSI Mode requires the COMMIT Statement with explicit pass-through.
- Whether or not to include nulls in the transformation INCLUDE_NULLS. Valid values are 'Yes' and 'No'. Default is 'No'.

TD_UNPIVOT allows you to transpose columns with same data type from wide to long. In the example, all sales columns sales_q1, sales_q2, sales_q3, sales_q4 and all cost columns cost_q1, cost_q2, cost_q3, cost_q4 are decimal data type.

When transposing columns with different data types, TD_UNPIVOT method will generate the error “Teradata execute: Failure in TD_Unpivot contract function. Error determining column type of value columns”. In this case you may consider the transposition solution presented in the following section. The solution is applicable to similar data transposition problem as discussed in this paper and is not intended to resolve all wide-to-long table transposition problems. However, understanding the transposition problem and the solution in this paper will help you create your own solution.

SOLUTION %WLTRANSPOSE

TD_UNPIVOT does not help transpose columns having a mixture of data types. The macro solution **`%wltranspose`** is introduced to address transposition of columns involving different data types. In Appendix A the definition of the macro can be found.

HIGHLIGHTS OF SOLUTION %WLTRANSPOSE

- is a MACRO solution consisting of 5 macro parameters
  - DSIN – input table for wide-to-long transposition;
  - ID – id column in the input table, e.g. NAME in examples below;
  - DSOUT – output table containing transposed data;
  - COLA – column categorizing rows in output table, e.g. COLUMNS in examples below;
  - COLB – column containing transposed data in output table, e.g. COL_VAL in examples below;
- is capable of transposing columns in INTEGER, DECIMAL, DATE, TIMESTAMP, VARCHAR, CHARACTER data types;
- is capable of creating the character column COL_VAL with a max length of 500 characters long in the output;
- creates two temporary process DBMS tables TRANS_TMP1 and TRANS_LEFT. Both tables are automatically checked and deleted before their creation in DBMS;
- is flexible for user to specify column names in the transposed output table. COLUMNS and COL_VAL are used in the solution and can be customized;

EXAMPLES OF USING THE SOLUTION

The following three examples demonstrate how wide-to-long table transposition is accomplished using the macro solution **`%wltranspose.`**

- Example 1 shows wide-to-long transposition involving integer column and character columns.
- Example 2 shows wide-to-long transposition involving integer column, character columns and date column.
- Example 3 shows wide-to-long transposition involving integer column, decimal column, float column character columns, date column and datetime column.

Example 1 Mixture of NUM and CHAR

Input Table

Name	Sex	Age	City
Thomas	M	11	New York
Mary	F	15	Los Angeles
Henry	M	14	Chicago
Alice	F	13	Houston

Display 3. Input Table With Integer and Character Columns

Output Table

Name	columns	COL_VAL
Alice	Age	13
Alice	City	Houston
Alice	Sex	F
Henry	Age	14
Henry	City	Chicago
Henry	Sex	M
Mary	Age	15
Mary	City	Los Angeles
Mary	Sex	F
Thomas	Age	11
Thomas	City	New York
Thomas	Sex	M

Display 4. Transposed Output Table

Using the macro solution described in the appendix A and calling the macro with the right input to the parameters DSIN, ID and DSOUT to get the transposed output table.

```
%wltranspose(dsin=display3, id=name, dsout=transposed_display3)
```

Example 2 Mixture of NUM, CHAR and DATE

Input Table

Name	Sex	Age	City	dt
Thomas	M	11	New York	26JAN2016
Mary	F	15	Los Angeles	26JAN2016
Henry	M	14	Chicago	26JAN2016
Alice	F	13	Houston	26JAN2016

Display 5. Input Table With Integer, Character And Date Columns

Output Table

Name	columns	COL_VAL
Alice	Age	13
Alice	City	Houston
Alice	dt	26Jan2016
Alice	Sex	F
Henry	Age	14
Henry	City	Chicago
Henry	dt	26Jan2016
Henry	Sex	M
Mary	Age	15
Mary	City	Los Angeles
Mary	dt	26Jan2016
Mary	Sex	F
Thomas	Age	11
Thomas	City	New York
Thomas	dt	26Jan2016
Thomas	Sex	M

Display 6. Transposed Output Table

Using the macro solution described in the appendix A and calling the macro with the right input to the parameters DSIN, ID and DSOUT to get the transposed output table.

```
%wltranspose(dsin=display5, id=name, dsout=transposed_display5)
```

Example3 Mixture of INTEGER, DECIMAL, FLOAT, CHAR, DATE and DATETIME

Input Table

Name	Sex	Age	City	dt	dtt0	Height	Weight
Thomas	M	11	New York	26JAN2016	26JAN2016:00:00:00	57.51	84.89
Mary	F	15	Los Angeles	26JAN2016	26JAN2016:00:00:00	66.53	111.75
Henry	M	14	Chicago	26JAN2016	26JAN2016:00:00:00	63.52	102.54
Alice	F	13	Houston	26JAN2016	26JAN2016:00:00:00	56.58	85.12

Display 7. Input Table With Integer, Decimal, Float, Character, Date And Datetime Columns

Output Table

Name	columns	COL_VAL
Alice	Age	13
Alice	City	Houston
Alice	dt	26Jan2016
Alice	dtt0	2016-01-26 00:00:00
Alice	Height	56.58
Alice	Sex	F
Alice	Weight	85.12
Henry	Age	14
Henry	City	Chicago
Henry	dt	26Jan2016
Henry	dtt0	2016-01-26 00:00:00
Henry	Height	63.52
Henry	Sex	M
Henry	Weight	102.54
Mary	Age	15
Mary	City	Los Angeles
Mary	dt	26Jan2016
Mary	dtt0	2016-01-26 00:00:00
Mary	Height	66.53
Mary	Sex	F
Mary	Weight	111.75
Thomas	Age	11
Thomas	City	New York
Thomas	dt	26Jan2016
Thomas	dtt0	2016-01-26 00:00:00
Thomas	Height	57.51
Thomas	Sex	M
Thomas	Weight	84.89

Display 8. Transposed Output Table

Using the macro solution described in the appendix A and calling the macro with the right input to the parameters DSIN, ID and DSOUT to get the transposed output table.

```
%wltranspose(dsin=display7, id=name, dsout=transposed_display7)
```

CONCLUSION

When transposing large DBMS tables from wide-to-long format, it is better to push the work to the DBMS side using SQL explicit pass-through. This paper presents a possible solution for performing the transpose work in the Teradata database. TD_UNPIVOT enables you to transpose table columns in the same data type. When you need to transpose a DBMS table which has a mixture of column data types, you could apply the solution in this paper. Once you understand the solution you should be able to customize it to address your specific transposition problems.

REFERENCES

Teradata Corporation. Teradata Database, Tools and Utilities Release 15.00. http://www.info.teradata.com/htmlpubs/DB_TTU_15_00/index.html#page/SQL_Reference/B035_1145_015K/Table_Operators.086.43.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author name	:	Tao Cheng
Company	:	Accenture Technology Solutions Netherlands
Work phone	:	+31 (0) 36 5461470
E-mail	:	t.cheng@accenture.com
Website	:	www.accenture.nl

APPENDIX A: SOLUTION %WLTRANSPOSE

```
/*common options to show in log the execution of codes in DBMS*/
option msglevel=n nostsuffix sastrace=',,,ds' sastraceloc=saslog
sql_ip_trace=source dbidirectexec;

/*declaring macro variables*/
%let connect_string =server=testserver user=testuser password=testpass;
%let td_db =testdatabase;

LIBNAME td TERADATA  &connect_string. schema=&td_db.  dbcommit=0;

%MACRO wltranspose(dsin=,id=,dsout=,cola=columns,colb=COL_VAL,len=VARCHAR(5
00));
  ***delete possible temporary process tables before creating them;
  %if %sysfunc(exist(td.trans_tmp1)) %then %do;
  PROC SQL;
    DROP TABLE td.trans_tmp1;
  QUIT;
  %end;
  %if %sysfunc(exist(td.trans_left)) %then %do;
  PROC SQL;
    DROP TABLE td.trans_left;
  QUIT;
  %end;
  %if %sysfunc(exist(td.&dsout.)) %then %do;
  PROC SQL;
    DROP TABLE td.&dsout.;
  QUIT;
  %end;
  ***create macro variables;
  DATA _NULL_;
    /*create macro variables - id column with single quotes*/
    id_sq = "'" || STRIP("&id.")|| "'";
    CALL SYMPUTX( 'idsq', id_sq);
    /*create macro variables - input dataset name with single quotes*/
    dsin_sq = "'" || STRIP("&dsin.")|| "'";
    CALL SYMPUTX( 'dsinsq', dsin_sq);
  RUN;
  ***create a table listing columns for wide-to-long transposition;
  PROC SQL;
   CONNECT TO teradata AS td (&connect_string.);
   EXECUTE(
           CREATE MULTISET TABLE &td_db..trans_tmp1 AS
           (
            SELECT TableName,ColumnName,ColumnFormat,ColumnType
            FROM DBC.COLUMNS
            WHERE UPPER(TableName) = UPPER(&dsinsq.) AND UPPER(ColumnName)
^=UPPER(&idsq.)
           )WITH DATA NO PRIMARY INDEX
          ) by td;
   EXECUTE(COMMIT) by td;
   DISCONNECT FROM td;
  QUIT;
```

```
  ***create macro variables;
  DATA _NULL_;
    SET td.trans_tmp1 end=last;
    n=STRIP(PUT(_N_, 8.));
    /*create macro variables - number of variables*/
    if last then do;
      CALL SYMPUTX("num", STRIP(PUT(_N_, 8.)));
    end;
    /*create macro variables - column types*/
    type_sq="'" || STRIP(ColumnType) ||"'";
    CALL SYMPUTX('vartp_' || n ,type_sq);
    /*create macro variables - column names*/
    CALL SYMPUTX('var_' || n ,strip(ColumnName));
    /*create macro variables - column names with single quotes*/
    name_sq="'" || STRIP(ColumnName) ||"'";
    CALL SYMPUTX('varsq_' || n ,name_sq);
    /*create macro variables - number of digits to the right of decimal
point*/
    if ColumnType='D' then len=length(strip(scan(ColumnFormat, 2, '.')));
    call symputx('dec_' || n , STRIP(PUT(len, 8.)));
  RUN;
***create the base table for table transposition in the following step;
  PROC SQL;
    CONNECT TO teradata AS td (&connect_string.);
    EXECUTE(
          CREATE MULTISET TABLE &td_db..trans_left AS (
            SELECT t1."&id.", t2.ColumnName as "&cola."
            FROM &td_db..&dsin. t1, &td_db..trans_tmp1 t2
          )WITH DATA NO PRIMARY INDEX
         ) by td;
    EXECUTE(COMMIT) by td;
    DISCONNECT FROM td;
  QUIT;
***create the transposed output dataset;
  PROC SQL;
    CONNECT TO teradata AS td (&connect_string.);
    EXECUTE(
          CREATE MULTISET TABLE &td_db..&dsout. AS (
            SELECT t1."&id.",
                   t1."&cola.",
                   CASE "&cola." %DO i=1 %TO &num.;
                   WHEN &&varsq_&i. %IF &&vartp_&i. = 'I'  %THEN THEN
CAST(CAST(t2.&&var_&i. AS INTEGER) AS &len.);
  %ELSE %IF &&vartp_&i. = 'F' %THEN THEN CAST(CAST(t2.&&var_&i. AS
DECIMAL(10,2)) AS &len.);
  %ELSE %IF &&vartp_&i. = 'D' %THEN THEN CAST(CAST(t2.&&var_&i. AS
DECIMAL(20,&&dec_&i.)) AS &len.);
  %ELSE %IF &&vartp_&i. = 'TS'  %THEN THEN CAST(t2.&&var_&i. AS &len.);
  %ELSE %IF &&vartp_&i. = 'CF' OR &&vartp_&i. = 'CV' %THEN THEN
t2.&&var_&i.;
  %ELSE %IF &&vartp_&i. = 'DA'  %THEN THEN CAST(CAST(t2.&&var_&i. AS DATE
FORMAT 'DDMMMYYYY') AS &len.);
  %END; END AS "&colb."
            FROM &td_db..trans_left t1 LEFT JOIN &td_db..&dsin. t2
            ON t1."&id." = t2."&id."
          )WITH DATA PRIMARY INDEX ("&id.","&cola.")
       ) by td;
  EXECUTE(COMMIT) by td;
  DISCONNECT FROM td;
  QUIT;
%MEND wltranspose;
```

APPENDIX B: EXAMPLE 1 INPUT TABLE

```
proc sql;
  connect to teradata (SERVER=testserver USER=testuser PASSWORD=testpass);
  execute (create table testdatabase.display3 (Name varchar(10),
                                                Sex varchar(1),
                                                Age integer,
                                                City varchar(30)
                                               )
          ) by teradata;
  execute (commit) by teradata;
  /*record 1*/
  execute (insert into testdatabase.display3 (Name, Sex, Age, City)
           values ('Thomas','M','11','New York')
          ) by teradata;
  /*record 2*/
  execute (insert into testdatabase.display3 (Name, Sex, Age, City)
           values ('Mary','F','15','Los Angeles')
          ) by teradata;
  /*record 3*/
  execute (insert into testdatabase.display3 (Name, Sex, Age, City)
           values ('Henry','M','14','Chicago')
           ) by teradata;
  /*record 4*/
  execute (insert into testdatabase.display3 (Name, Sex, Age, City)
           values ('Alice','F','13','Houston')
           ) by teradata;
  execute (commit) by teradata;
quit;
```

APPENDIX C: EXAMPLE 2 INPUT TABLE

```
proc sql;
  connect to teradata (SERVER=testserver USER=testuser PASSWORD=testpass);
  execute (create table testdatabase.display5 (Name varchar(10),
                                                Sex varchar(1),
                                                Age integer,
                                                City varchar(30),
                                                dt date
                                                )
          ) by teradata;
  execute (commit) by teradata;
  /*record 1*/
  execute (insert into testdatabase.display5 (Name, Sex, Age, City, dt)
           values ('Thomas','M','11','New York', '26JAN2016'(date,format
'ddmmmyyyy'))
           ) by teradata;
  /*record 2*/
  execute (insert into testdatabase.display5 (Name, Sex, Age, City, dt)
           values ('Mary','F','15','Los Angeles', '26JAN2016'(date,format
'ddmmmyyyy'))
           ) by teradata;
  /*record 3*/
  execute (insert into testdatabase.display5 (Name, Sex, Age, City, dt)
           values ('Henry','M','14','Chicago',  '26JAN2016'(date,format
'ddmmmyyyy'))
           ) by teradata;
  /*record 4*/
  execute (insert into testdatabase.display5 (Name, Sex, Age, City, dt)
           values ('Alice','F','13','Houston', '26JAN2016'(date,format
'ddmmmyyyy'))
           ) by teradata;
  execute (commit) by teradata;
quit;
```

APPENDIX D: EXAMPLE 3 INPUT TABLE

```
proc sql;
  connect to teradata (SERVER=testserver USER=testuser PASSWORD=testpass);
  execute (create table testdatabase.display7 (Name varchar(10),
                                               Sex varchar(1),
                                               Age integer,
                                               City varchar(30),
                                               dt date,
                                               dtt0 timestamp(0),
                                               Height decimal(6,2),
                                               Weight float
                                              )
          ) by teradata;
  execute (commit) by teradata;
  /*record 1*/
  execute (insert into &td_db..&tbl. (Name, Sex, Age, City, dt, dtt0,
Height, Weight)
           values ('Thomas','M','11','New York', '26JAN2016'(date,format
'ddmmmyyyy'), '26JAN2016:00:00:00' (timestamp(0), format
'ddmmmyyyybhh:mi:ss'), '57.51'(decimal(6,2), format'zzz9.99'), '84.89')
           ) by teradata;
  /*record 2*/
  execute (insert into &td_db..&tbl. (Name, Sex, Age, City, dt, dtt0,
Height, Weight)
           values ('Mary','F','15','Los Angeles', '26JAN2016'(date,format
'ddmmmyyyy'), '26JAN2016:00:00:00' (timestamp(0), format
'ddmmmyyyybhh:mi:ss'), '66.53'(decimal(6,2), format'zzz9.99'), '111.75')
           ) by teradata;
  /*record 3*/
  execute (insert into &td_db..&tbl. (Name, Sex, Age, City, dt, dtt0,
Height, Weight)
           values ('Henry','M','14','Chicago', '26JAN2016'(date,format
'ddmmmyyyy'), '26JAN2016:00:00:00' (timestamp(0), format
'ddmmmyyyybhh:mi:ss'), '63.52'(decimal(6,2), format'zzz9.99'), '102.54')
           ) by teradata;
  /*record 4*/
  execute (insert into &td_db..&tbl. (Name, Sex, Age, City, dt, dtt0,
Height, Weight)
           values ('Alice','F','13','Houston', '26JAN2016'(date,format
'ddmmmyyyy'), '26JAN2016:00:00:00' (timestamp(0), format
'ddmmmyyyybhh:mi:ss'), '56.58'(decimal(6,2), format'zzz9.99'), '85.12')
           ) by teradata;
  execute (commit) by teradata;
quit;
```

APPENDIX E: TERADATA COLUMN DATA TYPES

```
CF CHARACTER
F FLOAT
DA DATE
TS TIMESTAMP
I INTEGER
CV VARCHAR
A1 ARRAY
AN MULTI-DIMENSIONAL ARRAY
AT TIME
BF BYTE
BO BLOB
BV VARBYTE
CO CLOB
D DECIMAL
DH INTERVAL DAY TO HOUR
DM INTERVAL DAY TO MINUTE
DS INTERVAL DAY TO SECOND
DY INTERVAL DAY
HM INTERVAL HOUR TO MINUTE
HS INTERVAL HOUR TO SECOND
HR INTERVAL HOUR
I1 BYTEINT
I2 SMALLINT
I8 BIGINT
JN JSON
MI INTERVAL MINUTE
MO INTERVAL MONTH
MS INTERVAL MINUTE TO SECOND
N NUMBER
PD PERIOD(DATE)
PM PERIOD(TIMESTAMP WITH TIME ZONE)
PS PERIOD(TIMESTAMP)
PT PERIOD(TIME)
PZ PERIOD(TIME WITH TIME ZONE)
SC INTERVAL SECOND
SZ TIMESTAMP WITH TIME ZONE
TZ TIME WITH TIME ZONE
UT UDT Type
XM XML
YM INTERVAL YEAR TO MONTH
YR INTERVAL YEAR
++ TD_ANYTYPE
```