ISSN (online): 2250-141X
Vol. 8 Issue 1, March 2018

IMPLEMENTING PIVOT SOLUTIONS IN SAP HANA USING SQL SCRIPT

Venkat Ns Rao | Sr. Lead Consultant– Databases | TekLink Software Private Ltd.
Narayana Varma | Associate Director – SAP HANA & EDW |
TekLink Software Private Ltd.

Abstract**—***The Study or knowledge of Pivot structure in SAP HANA is necessary feature for developing this project. To understand Pivot this new script which generates pivot structure is useful. A pivot table is a data processing tool used to query, organize and summarize data or information between spreadsheets, tables or databases. Dragging and dropping fields into a pivot table facilitates rotational, or pivotal, structural changes. Pivot is not provided in SAP HANA as a standard function, hence this feature was developed using SQL script.*

Index Terms— Pivot Table; table; Join; Dynamic SQL;

I. INTRODUCTION

PIVOT rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output and performs aggregations where they are required on any remaining column values that are wanted in the final output. Pivoting is a common technique, especially for reporting, and it has been possible to generate pivoted result sets with SQL using Hana.

II. SOLUTION ABSTRACT

Business Case:
One of our client, a multinational company which is primarily into coating products was looking for a flexible and optimal solution based on SAP HANA for pricing waterfall analysis. As part of this solution we need to derive the values of various pricing item buckets such as rebates, commissions, discounts etc. for each of the billing items.
One of the challenging requirements in this entire solution is to convert the tabular data of invoice items into Pivot structure to show each pricing bucket in a separate column against the invoice items. The entire solution need to be dynamic since the exact columns in the Pivot results can change during time.

Implementing Pivot Structure in SAP HANA:

In specific scenarios such as the one we encountered in this business case, the solution need to be implemented to generate the pivot of the source data records to generate the required results. A pivot table is a data processing tool used to query, organize and summarize data or information between spreadsheets, tables or databases. Dragging and dropping fields into a pivot table facilitates rotational, or pivotal, structural changes, and pivot logic implemented using Dynamic SQL.

Dynamic SQL:

Dynamic SQL allows you to construct an SQL statement during the execution time of a procedure. While dynamic SQL allows you to use variables where they might not be supported in SQL Script and also provides more flexibility in creating SQL statements and Dynamic SQL statements are stored as

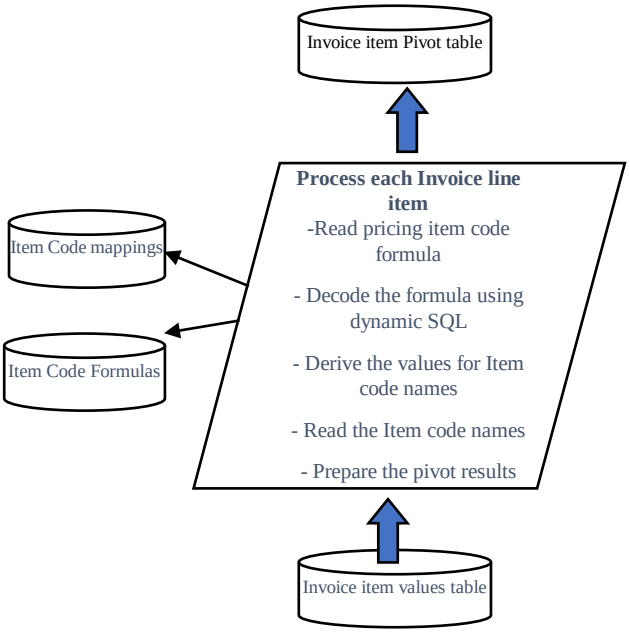
ISSN (online): 2250-141X
Vol. 8 Issue 1, March 2018

strings of characters that are entered when the program runs. They can be entered by the programmer or generated by the program itself, but unlike static SQL statements, they are not embedded in the source program. Also, in contrast to static SQL statements, dynamic SQL statements can change from one execution to the next.

Opportunities for optimizations are limited with Dynamic SQL.

- The statement is potentially recompiled every time the statement is executed.
- You cannot use SQLScript variables in the SQL statement.
- You cannot bind the result of a dynamic SQL statement to a SQLScript variable.

Solution process flow:



Algorithm for Pivot development script - Implemented using stored procedures based on SQL Script programming

- Define Cursor and take source table formula, and columns.
- Fetch formula and columns from source table into local variables.
- For some of the items value calculated using formula which is in **"ITEM_FORMULA_MAP" table** (for special discounts etc)**.**
- Read formula from **ITEM_FORMULA_MAP** table and remove all square braces using the **REPLACE** function in SQL and spilt formula comma separated string and count number of IDs(ITEM_ID) in formula using REPLACE function in code.
- If Number of IDs are less than one, then only one pricing item code is relevant and formula to calculate the target Pricing Item code is simple. For this we need to fetch the values from lookup table for Item code exists in formula and insert same value into look up table and construct SQL for final Pivot Table for target Item.
- If Number of IDs are more than one, in that case the target Pricing Item value should be calculated from multiple Item Values.
- If Number of IDs more than one then using cursor loop, construct SQL that will fetch Item invoice number and value for individual bucket and UNION all the values and add/subtract as per formula and insert into Pivot table using dynamic SQL.

Table Structure used for this development:

Input Data: Invoice line items with the pricing condition values
Table Structure:

```
CREATE COLUMN TABLE "ITEM_VALUES" (
"INVOICE_NUM" VARCHAR(30) CS_STRING NOT NULL
,"INVOICE_ITEM_NUM" VARCHAR(30) CS_STRING NOT NULL
,"ITEM_ID" SMALLINT CS_INT NOT NULL
,"TIME_STAMP" LONGDATE CS_LONGDATE
,"VALUE" DOUBLE CS_DOUBLE
,"RELEVANT_DATE" VARCHAR(8) CS_STRING
);
```

ISSN (online): 2250-141X
Vol. 8 Issue 1, March 2018

Sample Data:

	INVOICE_NUM	INVOICE_ITEM_NUM	ITEM_ID	TIME_STAMP	VALUE	RELEVANT_DATE
1	7711956831	000011	70	Jan 24, 2018 4:46:48.519 PM	13,776	20170309
2	7711956831	000061	70	Jan 24, 2018 4:46:48.519 PM	2,886	20170309
3	7711956831	000051	70	Jan 24, 2018 4:46:48.519 PM	2,076	20170309
4	7711956792	000031	70	Jan 24, 2018 4:46:48.519 PM	4,476	20170302
5	7711956831	000021	70	Jan 24, 2018 4:46:48.519 PM	3,509.7	20170309
6	7711956831	000022	70	Jan 24, 2018 4:46:48.519 PM	10,52...	20170309
7	7711956535	000031	70	Jan 24, 2018 4:46:48.519 PM	3,509.7	20170201
8	7711956641	000051	70	Jan 24, 2018 4:46:48.519 PM	1,819	20170215
9	7711956640	000021	70	Jan 24, 2018 4:46:48.519 PM	1,938...	20170215
10	7711956640	000111	70	Jan 24, 2018 4:46:48.519 PM	13,482	20170215
11	7711956640	000151	70	Jan 24, 2018 4:46:48.519 PM	23,41...	20170215
12	7711956640	000191	70	Jan 24, 2018 4:46:48.519 PM	684.8	20170215
13	7711956641	000021	70	Jan 24, 2018 4:46:48.519 PM	7,781...	20170215
14	7711956641	000061	70	Jan 24, 2018 4:46:48.519 PM	2,247	20170215
15	7711956641	000081	70	Jan 24, 2018 4:46:48.519 PM	2,247	20170215

Reference Data (Look up tables):

1) Pricing Item code and description mappings

Item Look up table

```
CREATE COLUMN TABLE "ITEM_NAMES" (
        "ITEM_ID"     VARCHAR(30)    NOT NULL
        "ITEM_NAME" VARCHAR(30)    NOT NULL
        );
```

Sample Data

ITEM_ID	ITEM_NAME
10	SUGGESTED_END_USER_PRICE
20	DISTRIBUTOR_ADJUSTMENT
30	DISTRIBUTOR_LIST_PRICE
40	REGIONAL_ADJUSTMENT
50	SEGMENT_ADJUSTMENT
60	CHANNEL_ADJ
70	GROSS_INVOICE_PRICE
80	COMPETITIVE_PRICE_DISCOUNT
90	NEGOTIATED_DISCOUNT
100	VOLUME_DISCOUNT
110	ORDER_QUANTITY_DISCOUNT
120	OFF_SPEC_DISCOUNT
130	ORDER_QTY_SURCHARGE
140	TRANSPORTATION_SURCHARGE
150	HANDLING_SURCHARGE
160	ENERGY_SURCHARGE
161	COMMODITY_SURCHARGE

2) Pricing Item Formula definitions

Table Structure:

```
        CREATE        COLUMN        TABLE "ITEM_FORMULA_MAP" (
        "ORG" VARCHAR(100) NOT NULL
        ,"TARGET_ITEM"      VARCHAR(50) NOT NULL
        ,"ITEM_FORMULA" VARCHAR(5000) NOT NULL
        ) UNLOAD PRIORITY 5 AUTO;
```

Sample Data:

	ORG	TARGET_ITEM	ITEM_FORMULA
1	SINGAPORE	50	[70]-[30]
2	SINGAPORE	10	[30] * 1.25
3	SINGAPORE	20	[30] * 0.25*-1

Output Table (Pivot results):

```
        CREATE        COLUMN        TABLE "ITEM_PIVOT" (
        "INVOICE_NUM"      VARCHAR(30) NOT NULL
        ,"INVOICE_ITEM_NUM" VARCHAR(30) NOT NULL
        ,"VDATE_BILLING_DATE" NVARCHAR(8)
        ,"REGION" VARCHAR(500)
        ,"SUGGESTED_END_USER_PRICE" DOUBLE CS_DOUBLE DEFAULT 0
        ,"DISTRIBUTOR_LIST_PRICE" DOUBLE CS_DOUBLE
        ,"DISTRIBUTOR_ADJUSTMENT" DOUBLE CS_DOUBLE
        ,"SEGMENT_ADJUSTMENT" DOUBLE CS_DOUBLE
        ,"GROSS_INVOICE_PRICE" DOUBLE CS_DOUBLE DEFAULT 0
        ,"SAMPLE_DISCOUNT"      DOUBLE CS_DOUBLE
        ,"GROSS_INVOICE_PRICE" DOUBLE CS_DOUBLE DEFAULT 0
        ,"REBATES" DOUBLE CS_DOUBLE
        ,"COMMISSIONS"          DOUBLE CS_DOUBLE
        ,PRIMARY KEY (
                "INVOICE_NUM"
                ,"INVOICE_ITEM_NUM"
                ,"REGION"
                )
        ) UNLOAD PRIORITY 5 AUTO
```

ISSN (online): 2250-141X
Vol. 8 Issue 1, March 2018

INVOICE_NUM	INVOICE_ITEM_NUM	VDATE_BILLING_DATE	REGION	SUGGESTED_END_USER_PRICE	DISTRIBUTOR_LIST_PRICE
7720673007	000531	20171122	SINGAPORE	2,047.5	1,638
7720672575	000141	20170215	SINGAPORE	1,575	1,260
7720673007	000041	20171122	SINGAPORE	11,756.25	9,405
7720672902	000162	20170919	SINGAPORE	370.5	296.4
7720672575	000631	20170215	SINGAPORE	786.5999999999999	629.28
7720672575	000101	20170215	SINGAPORE	148.275	118.62
7720673007	000962	20171122	SINGAPORE	1,100.925	880.74
7720672575	000572	20170215	SINGAPORE	236.9	189.52
7720672902	000171	20170919	SINGAPORE	1,214.4	971.52
7720672902	000661	20170919	SINGAPORE	115.1	92.08
7720672902	001089	20170919	SINGAPORE	799.05	639.24
7720673007	000541	20171122	SINGAPORE	3,247.5	2,598
7720673007	000051	20171122	SINGAPORE	1,837.5	1,470
7720672575	000151	20170215	SINGAPORE	270	216
7720673007	000551	20171122	SINGAPORE	2,722.5	2,178
7720672575	000602	20170215	SINGAPORE	535.5	428.4

EXAMPLE FOR ALGORITHM:

- Read item code name for 10 and formula for item code i.e. [30]*1.25
- Get the name for item code 10 and 30 from look up table i.e. item code 10 name is "Suggested end user price" and item code 30 is "Distributor list price"
- For one invoice and item level take value from formula item and update value using formula into target item for that invoice num at item level.
- Ex: Invoice num : `7720672575` and Invoice_item_num : `000141` for Singapore region

  Item 10 (suggested end user price) formula is **item 30 * 1.25** i.e. (distributor list price *1.25)

  Get the value for item 30 i.e. "distributor list price" from invoice item values table as shown below for above invoice number and invoice item number.

	INVOICE_NUM	INVOICE_ITEM_NUM	VALUE
1	7720672575	000141	1,260

So, for suggested end user price value as per formula is **distributor list price *1.25**

I.e. 1260*1.25 =1575 and same value is updated in Item Pivot table for that invoice number and invoice item. Similarly, the values are derived all the invoice items for remaining items values updated with the same algorithm.

III. CONCLUSION

Understanding of how Dynamic SQL works will be deeepened with clear explanations in this white paper and algorithms. You will be altered to potential performance problems that are not mentioned in the documentation and you will expand your repertoire of tuning solutions and troubleshooting techniques by learning how to use numerous hidden parameters and other undocumented features.

ACKNOWLEDGMENT

We would like to thank for our organization to help their technical support and facilitate lab frequently.

REFERENCES

I. https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx
II. https://help.sap.com/viewer/de2486ee947e43e684d39702027f8a94/2.0.02/en-US/966714d37630404983e8f4e3708ae79c.html
III. https://blogs.sap.com/2014/12/16/using-array-as-internal-table-to-handle-and-process-data/