Import Rules for Financial Consolidation

Contents

Import rules

1. Introduction

In the import definition, the import rules tab allows to apply transformations on imported data.

Typical transformations are, but are not limited to, :

- Removing rows not useful for the transfer of data
- Change the content of some fields based on some information
- Add some contents based on some other information
- …

Some information about import rules:

- Rules are part of the import structure definition (so included in backup/restore of them)
- Rules are based on pure SQL language.
- Applicable to any import (files / HUB / Odata).
- Reference to the imported data table made by using the variable $import$
- Possibility to make multiple transformation in the same script.
- For security reason, the rules are run under another context than the application, making it not possible to change other tables /structures of the application.
- The rules are accessible by the customer/consultant, so modifiable without requiring direct access on servers.
- Rules are applied on row data, before mapping and aggregation.

Note that the checkbox should be checked for the rule to be applied.

2. Import rule samples

Below some sample of rules.

Note that the import rules may contain multiple lines, in which case one transformation will be applied after the previous one.

Please remember that the import rules are processed before applying the mapping tables. The information available are the one of the file /hub before any conversion.

Change account code

- For all accounts (assumed on 6 positions) starting with '70' and having an intercompany detail information, replace the original account by taking the first 3 positions and putting '500' as the 3 last positions:

```
update $import$  set Account=left(Account, 3)+'500' where  ISNULL
(Partner, '') <> ''  AND left(Account, 2) = '70'
```

- Change account code if the amount is negative

```
update $import$  set Account='N'+Account where  left(Account, 2) =
'55' AND NetAmount < 0
```

Clearing the content of one field

```
Update $import$ set Partner = NULL where Partner = '#'
```

To load Dimension details and closing data in one go:

```
-- Copy all rows to temporary table
Select * into #tmp_no_dim from $import$

-- Update dimgroupcode with DIM (dim is mapped to the costcenter
details) in original rows
Update $import$ set DimensionGroup='DIM'

-- Remove all balance sheet accounts from import table because BS
does not have dimensions
-- delete from $import$ where left (Account,1) in ('0','1','2','6')
delete from $import$ where DimensionDetail1=0

 -- remove all dimension information from temp
Update #tmp_no_dim set DimensionDetail1 = null

-- copy tmp table to import table
Insert into $import$ select * from #tmp_no_dim

-- Drop temp tables
Drop table #tmp_no_dim
```

IC mapping, when the GL is equal but only some accounts are different across entities (mostly IC)

```
update $import$ set Partner='E20' where account='10080' and
company='E40'
update $import$ set Partner='E30' where account='210_E20' and
company='E20'
update $import$ set Partner='E40' where account='1550_E20' and
company='E20'
update $import$ set Partner='E20' where account='200_E10' and
company='E10'
update $import$ set Partner='E20' where account='1550_E10' and
company='E10'
update $import$ set Partner='E20' where account='1590_E10' and
company='E10'
update $import$ set Partner='E20' where account='1591_E10' and
company='E10'
update $import$ set Partner='E20' where account='1592_E10' and
company='E10'
```

Splitting data into different accounts according to an allocation key

```
/**
SAMPLE VALUES FROM IMPORT

...  ACCOUNT   CUSTOMTEXT2   NETAMOUNT ...

...  '6061100', 'P200', 1000.00
...  '6061105', 'P200', 9000.00

**/


-- Creating mapping table with allocation keys
DECLARE @MappingUnit1 table( Account nvarchar(20), CustomText2
nvarchar(20),Suffix nvarchar(4), Repartition decimal(6,2) )

-- Inserting the keys per account + Custom text 2
Insert into @MappingUnit1(Account, CustomText2, Suffix, Repartition)
VALUES ('6061100', 'P200', 'Adm',  0.4)
Insert into @MappingUnit1(Account, CustomText2, Suffix, Repartition)
VALUES ('6061100', 'P200', 'CopI', 0.6)

/* ...   */

-- Keeping a copy of original account as CustomText1
Update $import$ Set CustomText1 = Account
```

```
-- Create working table
DECLARE @TempImport table(
 RowNr int
, RowData nvarchar(max)
, ConsoID int
, Company nvarchar(20)
, Account nvarchar(20)

, NetAmount decimal(24,6)
, CustomText1 nvarchar(20)
, CustomText2 nvarchar(20)
)

-- inserting allocated values from import
Insert into @TempImport(RowNr, RowData, ConsoID, Company, Account,
NetAmount, CustomText1, CustomText2)
Select
        a.RowNr
        , a.RowData
        , a.ConsoID
        , a.Company
        , ( a.Account + b.Suffix ) as Account
        , (a.NetAmount * b.Repartition) as NetAmount
        , a.account as CustomText1
        , a.CustomText2

From $import$ a
        inner join @MappingUnit1 b on (b.Account = a.Account and b.
CustomText2 = a.CustomText2)


-- Deleting rows to split
DELETE From $import$
        where Account in (Select Account from @MappingUnit1)

-- inserting splitted rows
INSERT INTO $import$(RowNr, RowData, ConsoID, Company, Account,
NetAmount, CustomText1, CustomText2)
        Select RowNr, RowData, ConsoID, Company, Account, NetAmount,
CustomText1, CustomText2
                From @TempImport



/**
SAMPLE OUTPUT VALUES FROM RULE

...  ACCOUNT   CUSTOMTEXT1 CUSTOMTEXT2   NETAMOUNT ...

...  '6061100Adm', '6061100' , 'P200',  400.00
```

```
...   '6061100CopI','6061100' , 'P200',  600.00
...   '6061105',    '6061105' , 'P200', 9000.00

**/
```

Splitting a field and use parts of it in multiple other fields

In this use case (coming from EVS), the issue is that multiple dimensions are sent as one field, but contains multiple dimensions code concatenated using one separator (which was here the ‘pipe’ |)

```
/*
Let's assume that the concatenated information was loaded on
CustomText1 which is a nvarchar(max), meaning up to 8000 positions

Information is like 'ABC|123|---||DEF'
(some position might be empty)

Let's assume that position 2 should go to dimension 1 and position 5
should go to dimension 2
*/

-- Copy original CustomText1 in CustomText2 to keep the source info
untouched
Update $import$
   Set CustomText2 = CustomText1

-- We then need to ignore position 1
Update $import$
   Set CustomText2 = RIGHT(CustomText2,LEN(CustomText2)-CHARINDEX('|',
CustomText2))

-- We then need to use position 2 for Dimension 1
Update $import$
   Set DimensionDetail1 = LEFT(CustomText2,CHARINDEX('|',CustomText2)
-1)
   , CustomText2 = RIGHT(CustomText2,LEN(CustomText2)-CHARINDEX('|',
CustomText2))

-- We then need to ignore position 3
Update $import$
   Set CustomText2 = RIGHT(CustomText2,LEN(CustomText2)-CHARINDEX('|',
CustomText2))

-- We then need to ignore position 4
Update $import$
   Set CustomText2 = RIGHT(CustomText2,LEN(CustomText2)-CHARINDEX('|',
CustomText2))

-- We then need to use position 5 for Dimension 2
Update $import$
   Set DimensionDetail2 = LEFT(CustomText2,CHARINDEX('|',CustomText2)
-1)
   , CustomText2 = RIGHT(CustomText2,LEN(CustomText2)-CHARINDEX('|',
CustomText2))
```

Importing 12 columns of closing amounts (forecast, budget, …)

With the Custom Amounts fields, it is possible to import multiple columns from a file using the import rules.

Here is an example.

On basis of an excel file like the following, where

- NetAmount would be based on Column N
- all CustomAmount… fields filled in with columns C to N

A	B	C	D	E	F	G	H	I	J	K	L	M	N
COMPANY	ACCOUNT	1	2	3	4	5	6	7	8	9	10	11	12
WIE	41040	-26.768,49	-54.526,89	-84.794,71	-111.022,61	-136.122,48	-167.884,31	-198.939,10	-233.194,42	-258.110,70	-283.784,52	-312.629,26	-346.031,86
WIE	41080	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
WIE	41090	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
WIE	41110	-12.121,60	-23.900,93	-35.785,53	-47.681,98	-59.787,43	-72.124,64	-89.261,67	-100.861,74	-107.434,48	-121.887,81	-132.430,49	-143.643,10

We could have a rule like the following to handle the 12 columns:

```
-- Copy all rows to temporary table
Select * into #tmp_budget from $import$

-- Keep Period information to be used later in the rule
DECLARE @Period nvarchar(12)
DECLARE @Year nvarchar(4)
DECLARE @PeriodSuffix nvarchar(6)

Select TOP 1 @Period = Period
   , @Year = LEFT(Period, 4)
   , @PeriodSuffix = RIGHT(Period, 6)
  from #tmp_budget
  Where ISNULL(Period, '') <> ''

-- Update Temporary Table to match the January period
Update   #tmp_budget
   Set NetAmount = CustomAmount1
   , Period = @Year + '01' + @PeriodSuffix

-- INSERT January data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Feb. period
Update   #tmp_budget
   Set NetAmount = CustomAmount2
   , Period = @Year + '02' + @PeriodSuffix

-- INSERT Feb. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Mar. period
Update   #tmp_budget
   Set NetAmount = CustomAmount3
   , Period = @Year + '03' + @PeriodSuffix
```

```
-- INSERT Mar. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Apr. period
Update   #tmp_budget
   Set NetAmount = CustomAmount4
   , Period = @Year + '04' + @PeriodSuffix

-- INSERT Apr. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the May period
Update   #tmp_budget
   Set NetAmount = CustomAmount5
   , Period = @Year + '05' + @PeriodSuffix

-- INSERT May. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Jun. period
Update   #tmp_budget
   Set NetAmount = CustomAmount6
   , Period = @Year + '06' + @PeriodSuffix

-- INSERT Jun. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Jul. period
Update   #tmp_budget
   Set NetAmount = CustomAmount7
   , Period = @Year + '07' + @PeriodSuffix

-- INSERT Jul. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Aug. period
Update   #tmp_budget
   Set NetAmount = CustomAmount8
   , Period = @Year + '08' + @PeriodSuffix

-- INSERT Aug. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Sep. period
Update   #tmp_budget
```

```
   Set NetAmount = CustomAmount9
   , Period = @Year + '09' + @PeriodSuffix

-- INSERT Sep. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Oct. period
Update   #tmp_budget
   Set NetAmount = CustomAmount10
   , Period = @Year + '10' + @PeriodSuffix

-- INSERT Oct. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Nov. period
Update   #tmp_budget
   Set NetAmount = CustomAmount11
   , Period = @Year + '11' + @PeriodSuffix

-- INSERT Nov. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- It is assumed that the NetAmount was initially matching the column
for december, so no need to add it once again
```

Importing 12 columns of MONTHLY values(forecast, budget, …)

On basis of previous use case, let's now assume that the columns don't contains year-to-date values, but monthly values.

In such cases, we could use the same code than above, but instead of setting the NetAmount to another Custom Value, we would add it

```
- Copy all rows to temporary table
Select * into #tmp_budget from $import$

-- Keep Period information to be used later in the rule
DECLARE @Period nvarchar(12)
DECLARE @Year nvarchar(4)
DECLARE @PeriodSuffix nvarchar(6)

Select TOP 1 @Period = Period
   , @Year = LEFT(Period, 4)
   , @PeriodSuffix = RIGHT(Period, 6)
  from #tmp_budget
  Where ISNULL(Period, '') <> ''

-- Remove all existing data in the import as it would probably not be
```

```
correct
DELETE FROM $import$

-- Update Temporary Table to match the January period
-- Monthly January = Year-to-date January
Update   #tmp_budget
   Set NetAmount = ISNULL(CustomAmount1, 0)  -- we handle the NULL by
replacing them by zero
   , Period = @Year + '01' + @PeriodSuffix

-- INSERT January data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Feb. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount2, 0)
-- we add the monthly value to existing amount
   , Period = @Year + '02' + @PeriodSuffix

-- INSERT Feb. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Mar. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount3, 0)
   , Period = @Year + '03' + @PeriodSuffix

-- INSERT Mar. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Apr. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount4, 0)
   , Period = @Year + '04' + @PeriodSuffix

-- INSERT Apr. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the May period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount5, 0)
   , Period = @Year + '05' + @PeriodSuffix

-- INSERT May. data into the import
INSERT INTO $import$
Select * from #tmp_budget
```

```
-- Update Temporary Table to match the Jun. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount6, 0)
   , Period = @Year + '06' + @PeriodSuffix

-- INSERT Jun. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Jul. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount7, 0)
   , Period = @Year + '07' + @PeriodSuffix

-- INSERT Jul. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Aug. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount8, 0)
   , Period = @Year + '08' + @PeriodSuffix

-- INSERT Aug. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Sep. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount9, 0)
   , Period = @Year + '09' + @PeriodSuffix

-- INSERT Sep. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Oct. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount10, 0)
   , Period = @Year + '10' + @PeriodSuffix

-- INSERT Oct. data into the import
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Nov. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount11, 0)
   , Period = @Year + '11' + @PeriodSuffix

-- INSERT Nov. data into the import
```

```
INSERT INTO $import$
Select * from #tmp_budget

-- Update Temporary Table to match the Dec. period
Update   #tmp_budget
   Set NetAmount = ISNULL(NetAmount, 0) + ISNULL(CustomAmount12, 0)
   , Period = @Year + '12' + @PeriodSuffix

-- INSERT Dec. data into the import
INSERT INTO $import$
Select * from #tmp_budget
```

3. Import Rules fields

For the transformation, the following fields can be used.

Fields in $import$ table:

Field name:	Size	Description
TmpImportDataID	int	
RowNr	int	
RowData	nvarchar(max)	
Period	nchar(12)	
ConsoID	int	
Company	nvarchar(20)	
Account	nvarchar(20)	
Flow	nvarchar(20)	
Partner	nvarchar(20)	
NetAmount	decimal(24,6)	
Debit	decimal(24,6)	
Credit	decimal(24,6)	
JournalEntry	int	
JournalDescription	nvarchar(120)	
HistoricalRate	decima(24,8)	
TransactionCurrency	nvvarchar(3)	
TransactionAmount	decimal(24,6)	
DimensionGroup	nvarchar(25)	
DimensionDetail1	nvarchar(25)	

DimensionDetail2	nvarchar(25)	
DimensionDetail3	nvarchar(25)	
HasConversionError (= 0)	bit	
TableName (NULL)	varchar(8)	
DoProcess (= 1)	bit	

Additional fields available as from v3.1:

Field name	Size	Description
CustomAmount1		
CustomAmount2		
CustomAmount3		
CustomAmount4		
CustomAmount5		
CustomAmount6		
CustomAmount7		
CustomAmount8		
CustomAmount9		
CustomAmount10		
CustomAmount11		
CustomAmount12		
CustomText1		
CustomText2		
CustomText3		
CustomText4		
CustomText5		
CustomText6		
CustomText7		
CustomText8		
CustomText9		
CustomText10		
CustomText11		
CustomText12		