ETL Approach for Transposing Data

Splitting Single Column Multi Values into Multiple Rows

Harish Kumar
FET, Rama University, Kanpur,
Uttar Pradesh, Kanpur, India
hc78@rediffmail.com

Sunil Kumar
Research Scholar, Deptt. CSE, Bhagwant University, Ajmer,
Research work carried out, Bank of America, USA
sunilyadavpmp@gmail.com

***Abstract*—**There are many real time scenarios where we need to split single column's multiple values separated by comma, tabs, whitespaces, other value separator symbols used at client file generation systems. The process for splitting single column values into multiple row/records can be achieved ETL Normalization transformation; by using unpivot database function or using expression transformation and splitting field values into multiple segments. In this article, I will explain the research work by using ETL Normalization transformation with Informatica 9.X tool, Oracle 10G database.

The process of organizing data into a database is called normalization. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency. The redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A client address change is much easier to implement if that data is stored only in the Client table and nowhere else in the database. Inconsistent dependencies can make data difficult to access because the path to find the data may be missing or broken. There are a few rules for database normalization. Each rule is called a "Normal Form." If the first rule is observed, the database is said to be in "first normal form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications. As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some Client find this cumbersome. If you decide to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

***Index Terms*—** Normalization, 1NF, 2NF, 3NF, ETL, Oracle 10G, DDL, DML

I. INTRODUCTION

The Normalizer transform can ease many complex data transformation requirements. The Normalizer transformation is active transformation in Informatica. Normalizer transformation can output multiple rows for each input row; it can transpose columns to rows as well. The Normalizer transformation parses multiple-occurring columns from COBOL sources, relational tables, or other sources. Normalizer can be used to transpose the data in columns to rows. In summary Normalizer effectively does the opposite of what Aggregator transformation functionality.

II. TRANSPOSING DATA USING NORMALIZER

Let us consider we have a source system raw data text file IC_U65_CMS_REG.txt that contains following 5 fields:

- NPN
- Marketplace
- RegistrationDate
- RegistrationYear
- RegistrationEndDate

Sample Data file contains 3 records as multiple NPN values separated by comma and each filed is tab separated as per following format:

TABLE I. SOURCE FILE DATA LAYOUT

```
NPN Marketplace RegistrationDate RegistrationYear RegistrationEndDate

1111111 Individual 7/26/2014 2014 12/31/2014

2222222,3333333 Both 7/26/2015 2015 12/31/2015

444444,5555555,666666 Both 7/26/2013 2013 12/31/2013
```

III. Expected Output After normalization

I will load text file **IC_U65_CMS_REG.txt** data into one of relational table called **bcbsf_ps_u65_ffm_reg_preload** table resides into Oracle Database schema. After loading above sample data target table **bcbsf_ps_u65_ffm_reg_preload** should look like as follows:

TABLE II. Target table after ETL load

NPN	MPlace	RDate	RYear	REDate
1111111	Individual	7/26/2014	2014	12/31/2014
2222222	Both	7/26/2015	2015	12/31/2015
333333	Both	7/26/2015	2015	12/31/2015
444444	Both	7/26/2013	2013	12/31/2013
5555555	Both	7/26/2013	2013	12/31/2013
666666	Both	7/26/2013	2013	12/31/2013

IV. ETL solution approach

We can achieve above output by using multiple ways such as by using unpivot database function, by writing each rows into some temporary tables then using expression transformation along with SUBSTR database function, by using Informatica Normalizer transformation. In this article, I will write solution by using Informatica Normalizer transformation to achieve the above output.

First of all the text file data to be transformed, cleansed then loaded into table bcbsf_ps_u65_ffm_reg_preload**.** The table has following Data Definition Language (DDL):

TABLE III. Table bcbsf_ps_u65_ffm_reg_preload Definition

```
CREATE TABLE bcbsf_ps_u65_ffm_reg_preload
    (stageicu65cmsregseq      NUMBER(*,0) NOT NULL,
     npn                      VARCHAR2(30 BYTE),
     marketplace              VARCHAR2(30 BYTE),
     registrationdate         DATE,
     registrationyear         VARCHAR2(4 BYTE),
     registrationenddate      DATE)
    PCTFREE    10
    INITRANS   1
    MAXTRANS   255
    TABLESPACE tc_sm_data
    STORAGE    (
     INITIAL    131072
     NEXT       262144
     PCTINCREASE 0
     MINEXTENTS  1
     MAXEXTENTS  2147483645 )
    NOCACHE
    MONITORING
    NOPARALLEL
    LOGGING;
```

Following roles, permissions and Grants to be assigned to this table:

```
GRANT SELECT ON bcbsf_ps_u65_ffm_reg_preload TO icomp_read_role;

GRANT DELETE ON bcbsf_ps_u65_ffm_reg_preload TO icomp_update_role;

GRANT INSERT ON bcbsf_ps_u65_ffm_reg_preload TO icomp_update_role;

GRANT SELECT ON bcbsf_ps_u65_ffm_reg_preload TO icomp_update_role;

GRANT UPDATE ON bcbsf_ps_u65_ffm_reg_preload TO icomp_update_role;
```

This table contains all 5 columns/field as available in source text file along with new field called stageicu65cmsregseq **.** This field stores the incremental sequence values to keep the data integrity and consistency.

V. Normalizer , ROUTER Transformation Property

As normalization transformation contains tab called Normalizer. This tab having following fields(Column Name, Level, Occurs, Datatype, Prec, Scale, Key Type) **Occurs** the property need to set the based on repetitive values of NPN captured in source system data. In this article we have set this property to 4. This means the ETL process will create total 6 records for each source system rows with GCID_NPN values for first source record will be 1, for second record (1,2), 3rd record (1,2,3) based on Occurs property set values as well NPN occurrences. The GK_NPN will be work as primary key for all

records generated during Normalizer process. In this case GK_NPN will have key values from 1 to 6

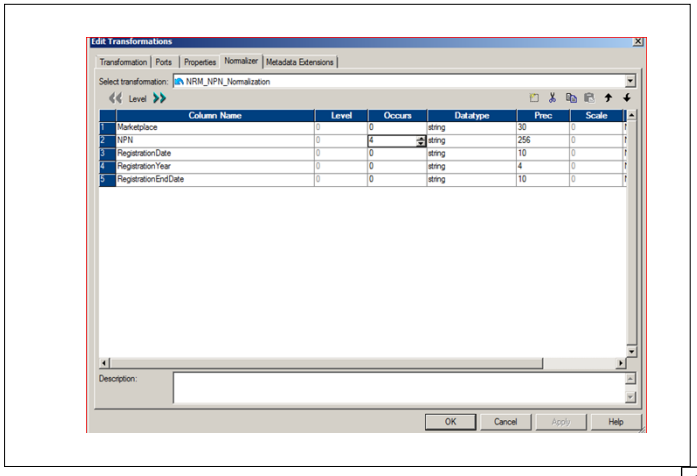
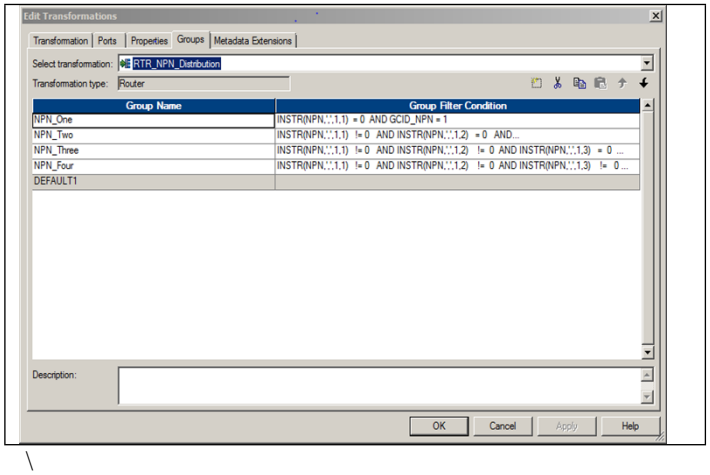


Fig. 1. Normalization transformation property setup

In mapping router transformation called *RTR_NPN_Distribution* will be created and 4 GROUPS will be created. Each source record's NPN values will be tested and Incoming data will incoming data will satisfy one of following group condition.



\

Fig. 2. Router transformation group condition setup

TABLE IV. ROUTER GROUP CONDITION

Group Name	Group Condition
NPN_one	INSTR(NPN,',',1,1) = 0 AND GCID_NPN = 1
NPN_Two	INSTR(NPN,',',1,1) != 0 AND INSTR(NPN,',',1,2) = 0 AND GCID_NPN = 2
NPN_Three	INSTR(NPN,',',1,1) != 0 AND INSTR(NPN,',',1,2) != 0 AND INSTR(NPN,',',1,3) = 0 AND GCID_NPN = 3
NPN_Four	INSTR(NPN,',',1,1) != 0 AND INSTR(NPN,',',1,2) != 0 AND INSTR(NPN,',',1,3) != 0 AND INSTR(NPN,',',1,4) = 0 AND GCID_NPN=4

VI. FINAL SOLUTION

This solution has been reached and implemented by using Informatica Normalizer function, along Oracle INSTR function. The INSTR functions searches a string for a substring using characters and returns the position in the string that is the first character of a specified occurrence of the substring. A nonzero INTEGER when the search is successful or 0 (zero) when search is unsuccessful. As the All NPN values are separated by commas therefore by using INSTR function we will get the position of each comma (,) after identifying each comma Oracle SUBSTR function will bifurcate the actual input string from beginning to first position of comma minus 1 for capturing only those records from source system where we have only single NPN from source system, as following will be applied logic to capture only records where only single NPN is input. The first record from source sample can be captured from below logic:

Logic to capture first source file record:

Expression *EXP_NPN_One* (RTRIM(LTRIM(NPN1))) will take input of rows that satisfy the *NPN_One* Group Condition

Logic to capture second source file record:

EXP_NPN_Two will take the input of rows that satisfy Router Group condition NPN_Two only.

This means *EXP_NPN_Two* will process only those records from source where 2 NPN values are coming from source system.

In this case second record {2222222,3333333 Both 7/26/2015 2015 12/31/2015 } from sample data will be written through *EXP_NPN_Two* expression.

Logic to capture and write this record will be as follows:

First of all declare one input string variable called as NPN_V1 to Store the non zero value of expression INSTR(NPN3,',',1,1)

NPN_V1 = INSTR(NPN3,',',1,1) =7 (n^{th} position of first occurrence of comma for second record)

Define 2 output variables NPN3_Out1, NPN3_Out2 to write two records for NPN second source record

This means the first NPN value 2222222 can be captured with following expression

NPN3_Out1 =RTRIM(LTRIM(SUBSTR(NPN3,1,NPN3_V1-1)))

Second NPN value 3333333 can be captured with following expression

NPN3_Out2 = LTRIM(SUBSTR(NPN3,NPN3_V1+2))

NPN4_V1 = INSTR(NPN4,',',1,1) = 7 (n^{th} position of first occurrence of comma for third record) , to calculate n^{th} position below mentioned query can be used .

Select INSTR('444444,5555555,666666',',',1,1) FROM DUAL;--- First Occurrence of comma Output will be 7

Select INSTR('444444,5555555,666666',',',1,2) FROM DUAL; --- Second Occurrence of comma Output will be 15

NPN4_V2 = INSTR(NPN4,',',1,2) = 15 (n^{th} position of Second occurrence of comma for third record)

Define 3 output variables NPN4_Out1, NPN4_Out2, NPN4_Out3 to write three records for NPN third source record

This means the first NPN value 444444 can be captured with following expression

NPN4_Out1=RTRIM(LTRIM(SUBSTR(NPN4,1,NPN4_V1-1)))

Second NPN value 5555555 can be captured with following expression

NPN4_Out2= RTRIM(LTRIM(SUBSTR(NPN4,NPN4_V1+1,(NPN4_V2-NPN4_V1-1))))

Third NPN value 666666 can be captured with following expression

NPN4_Out3= RTRIM(LTRIM(SUBSTR(NPN4,NPN4_V2+1)))

Similar logic can be applied for 4 NPN Values separated by three commas (11111,222222,3333333,4444444, Both 7/26/2016 2016 12/31/2016) in real word projects.

Logic to capture third source file record:

EXP_NPN_Three will process only those records from source where 3 NPN values are coming from source system.

In this case Third record {444444,5555555,666666 Both 7/26/2013 2013 12/31/2013} from sample data will be written through *EXP_NPN_Three* expression. Logic to capture and write this record will be as follows:

First of all declare two input string variables called as NPN4_V1, NPN4_V2 Store the non zero value of expressions

The ETL mapping design to research the proposed solution.

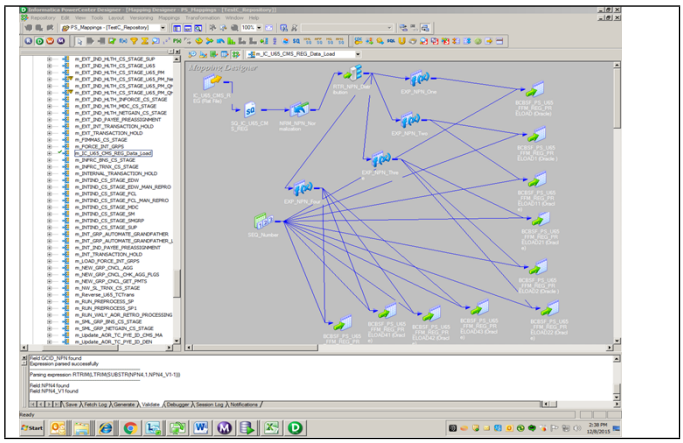


Fig. 3. ETL mapping design and flow

In each expression transformation other two date fields called as RegistrationDate, RegistrationEndDate are converted from STRING data type to Date Data type then each record is written to Oracle Database table bcbsf_ps_u65_ffm_reg_preload

The function LTRIM and RTRIM is used to remove any blank space/White space from source data input file.

VII. Output and Statistics

After the mapping design, validation and testing , One of session *s_m_IC_U65_CMS_REG_Data_Load* is created and called by workflow ***wf_ IC_U65_CMS_REG_Data_Load*** to run the session/mapping. Open and run the Workflow Manager tool to run this workflow. Session has written total 6 records into database table.

The expression EXP_NPN_One will pass and write only one records to target instance of BCBSF_PS_U65_FFM_REG_PRELOAD table. The expression EXP_NPN_Two will pass and write only two records to target instances (BCBSF_PS_U65_FFM_REG_PRELOAD1, CBSF_PS_U65_FFM_REG_PRELOAD11)of BCBSF_PS_U65_FFM_REG_PRELOAD table.

The expression EXP_NPN_Three will pass and write only three records to target instances (BCBSF_PS_U65_FFM_REG_PRELOAD2, BCBSF_PS_U65_FFM_REG_PRELOAD21, BCBSF_PS_U65_FFM_REG_PRELOAD2)of BCBSF_PS_U65_FFM_REG_PRELOAD table.

The expression EXP_NPN_Four will not pass and write any record to target BCBSF_PS_U65_FFM_REG_PRELOAD table. As there is no data to satisfy the router group 4 condition.

Partial session log has been captured to verify the result loaded by various instance of BCBSF_PS_U65_FFM_REG_PRELOAD table.

TABLE V. Table Session Load Summary

TM_6252 Source Load Summary.
Table: [SQ_IC_U65_CMS_REG] (Instance Name: [SQ_IC_U65_CMS_REG])
Output Rows [3], Affected Rows [3], Applied Rows [3], Rejected Rows [0]
TM_6253 Target Load Summary.
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD43])
Output Rows [0], Affected Rows [0], Applied Rows [0], Rejected Rows [0]
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD42])
Output Rows [0], Affected Rows [0], Applied Rows [0], Rejected Rows [0]
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD41])
Output Rows [0], Affected Rows [0], Applied Rows [0], Rejected Rows [0]
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD4])
Output Rows [0], Affected Rows [0], Applied Rows [0], Rejected Rows [0]
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD22])
Output Rows [1], Affected Rows [1], Applied Rows [1], Rejected Rows [0]
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD21])
Output Rows [1], Affected Rows [1], Applied Rows [1], Rejected Rows [0]
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD2])
Output Rows [1], Affected Rows [1], Applied Rows [1], Rejected Rows [0]
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD11])
Output Rows [1], Affected Rows [1], Applied Rows [1], Rejected Rows [0]
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD1])

Output Rows [1], Affected Rows [1], Applied Rows [1], Rejected Rows [0]
Table: [BCBSF_PS_U65_FFM_REG_PRELOAD] (Instance Name: [BCBSF_PS_U65_FFM_REG_PRELOAD])
Output Rows [1], Affected Rows [1], Applied Rows [1], Rejected Rows [0]
INFO 12/8/2015 3:01:04 PM node01_edcaincd201.bcbsfl.com 7455 LM_36318
Session task instance [s_m_IC_U65_CMS_REG_Data_Load]: Execution succeeded.
INFO 12/8/2015 3:01:06 PM node01_edcaincd201.bcbsfl.com 8483 LM_36318
Workflow [wf_IC_U65_CMS_REG_Data_Load]: Execution succeeded.

Open the Workflow Monitor tool to see the output of Workflow and Session Execution statistics

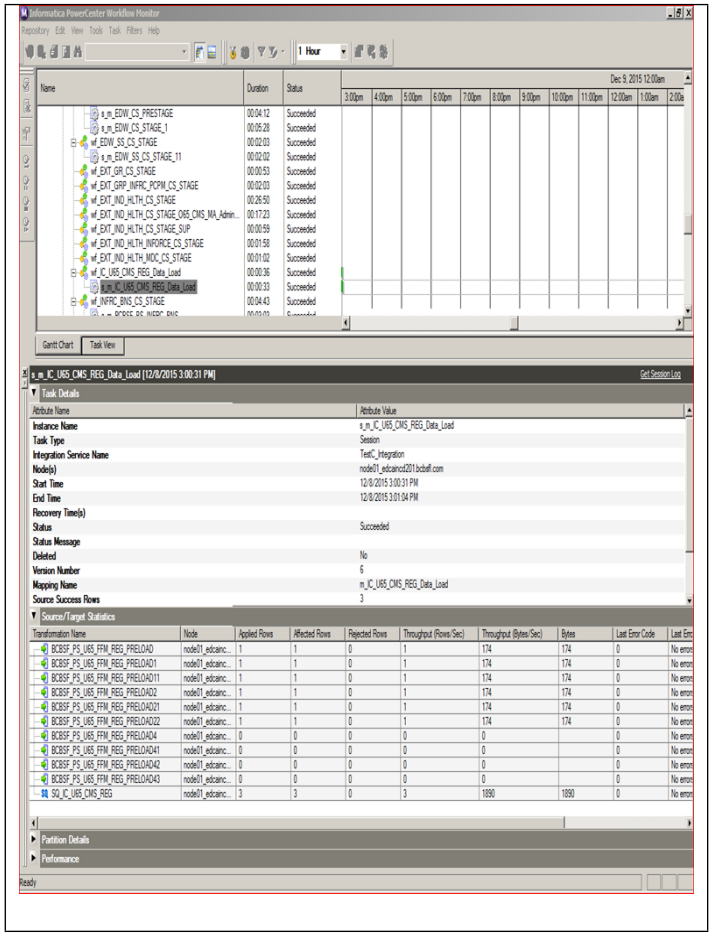


Fig. 4. Workflow execution statistics

After successful workflow execution run the following query into ICOMPTCO database TC1 schema to cross verify result set from Oracle database schema.

*Select * FROM bcbsf_ps_u65_ffm_reg_preload;-- 6 Rows loaded after workflow successful execution.*

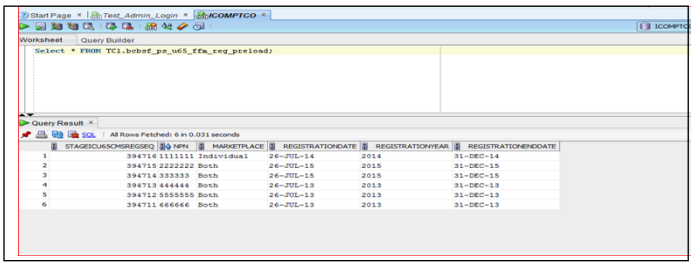


Fig. 5. Data result set after NPN Normalization

REFERENCES

[1] https://support.microsoft.com/en-us/kb/283878
[2] https://docs.oracle.com/cd/B19306_01/server.102/b14200/functions068.htm
[3] http://www.disoln.org/2012/10/Update-With-Out-Update-Strategy-for-Better-Session-Performance.html
[4] https://www.informatica.com/solutions.html#fbid=GWKrXmpNL9n
[5] http://www.techonthenet.com/oracle/functions/substr.php
[6] http://infatips.com/informatica_normalizer_transformation_-_levels_and_occurs/#.Vmhy9tLR9pg
[7] http://www.kimballgroup.com/2010/03/three-etl-compromises-to-avoid/
[8] https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=12490
[9] http://infatips.com/informatica_normalizer_transformation_-_levels_and_occurs/#.Vmhy9tLR9pg
[10] http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_app_sqlfunc.htm#OLADM930198