Redesigning a Flexible Material Master Data Application with Language Dependency

Mihaela Muntean
Department of Business Information Systems, West University of Timisoara, Bld. Vasile Parvan, no. 4, room 144, 300223, Timisoara, Romania
mihaela.muntean@e-uvt.ro

Cornelia Muntean
Department of Business Information Systems, West University of Timisoara, Bld. Vasile Parvan, no. 4, room 144, 300223, Timisoara, Romania
cornelia.muntean@e-uvt.ro

Abstract

Based on some inconveniences identified within a material master data application of the SAP MM module, the current paper proposes an alternative technical solution for designing the popup layout screen of the material master data application with respect to language dependency. The application should present the text of all tabs, areas and fields in the desired language based on the user logon preferences. Also it has been implemented a possibility to sort the data in the desired order, so the customer isn't forced, as before, to search for a certain field, when updating certain materials. The proposed solution suggests the use of five database tables, combined within three maintenance views, which build a view cluster. The advantage of the redesigned application consists of easier maintainability of the data (fields, areas and tabs can be easily added, deleted, reordered and renamed) and all the data within the view cluster can be translated into any language supported by the system.

Keywords: Material Master Data Application, language dependency, maintenance view, view cluster

1. Introduction

In the SAP MM module, material master data refers to all the material master records that are stored in the system. The material master record is used by different departments of an organization, and each department requires different information about the material (Johnson, 2013). The Material Master isn't just a single file but a number of tables of information that combined reflect all of the information for that material. The Material Master transaction is structured so that there are entry screens for different functional information such as Purchasing, Sales, or Accounting, but there is also an organizational dimension to data entry. The material information can be entered at each level of the organization, for example, at the levels of plant, storage location, or sales organization.

SAP systems are multilingual, maintenance in multiple languages being possible (Ashfaque, 2014). The syntagma *language dependency* refers to the fact that all text-elements in the user interface depend on the logon language. The logon language has to be automatically used for the description of all menus, pop-ups, tabs, areas, fields etc. Thus, global customizing and configuration texts, that must be available throughout the enterprise in all countries in a local language, must be translated. Because most of the customizing is customer - specific, almost all international customers need to translate this kind of data. Master data is an important area which requires translation. The most important group in this category is material and product data. Because a global enterprise offers identical products in many countries, the product descriptions must be translated into all the languages of these countries.

The SAP master data governance framework provides tools to ensure consistent master data along end-to-end business processes, including language management (Jacob, 2012). Most objects in the user interface are language-dependent, and the system uses resources tables/files to contain

the translations. The language dependency of the Material Master Data was modeled in a single database table, solution that needed to be reconsidered in order to gain flexibility. Our demarche consists of a technical solution to improve SAP Material Master Data language dependency, solution that nowadays is subject of the final testing to become operational within the SAP MM module.

2. Implementation of the SAP Material Master Data Application with language dependency

2.1. Previous implementation. Drawbacks.

The Material Master Data Application provides an update popup layout, including tabs, areas and fields (also customer-specific fields). Each tab consists of one or more areas and each area of one or more fields, similar to the example below (figure 1). The application is called flexible because the user must have the possibility to add, delete, reorder or rename tabs, areas and fields.

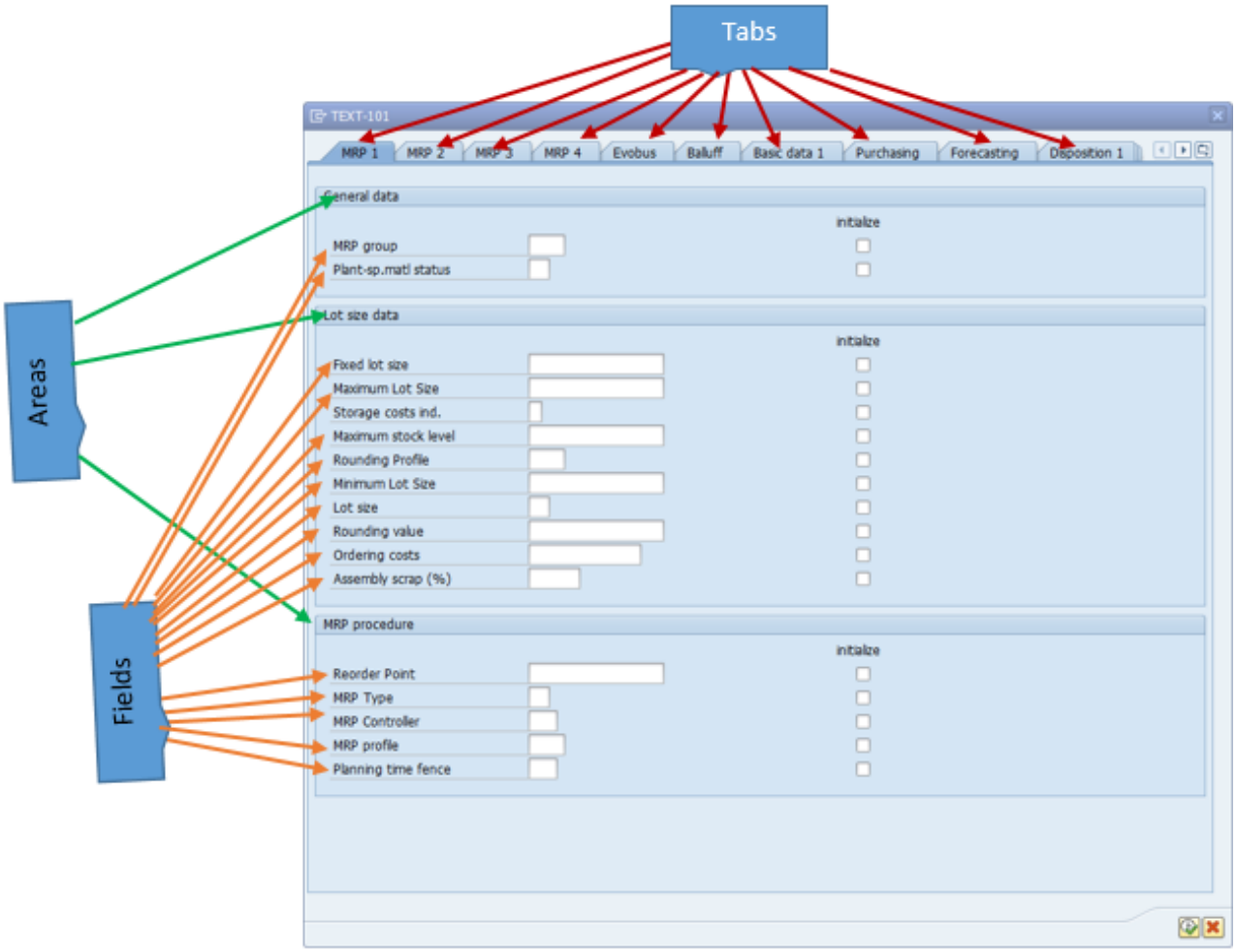

Figure 1. Popup layout for the Material Master Data Application

The correct update of the fields has to be ensured. For each field the user can choose between initializing it, updating it to a certain value or ignore it (by leaving it blank).

The layout should remain consistent throughout different languages, supporting the logon language of the user. The text of all tabs, areas and fields should be translated into the chosen language. However, it has to be possible to maintain the labels in different languages (Ong, 2014).

In the previous implementation there has been used a single database table which did not provide a consistent overview of existing tabs, areas and fields, as well as of the languages, in which a specific field of an area or tab was translated. So, it was difficult to maintain this database table by the customizing end-users in different languages, because every update of a tab, area or field

required a number of actions in this table which had to be done manually and very carefully, requiring much time and attention. The number of rows of this table was very large and the content looked like the one shown in figure 2.

	A	B	C	D	E	F
1	Table Name	Field Name	Language	Tab description	Area description	Level of Analys
17	DPOP	ALPHA	CS	Prognoza	Ridici data	D
18	DPOP	ALPHA	DE	Prognose	Steuerungsdaten	D
19	DPOP	ALPHA	EN	Forecast	Control data	D
20	DPOP	ALPHA	FR	Prevision	Donnees de pilotage	D
21	DPOP	ALPHA	IT	Previsione	Dati di controllo	D
22	DPOP	ALPHA	DA	Prognose	Styredata	D
23	DPOP	ALPHA	PL	Prognoza	Dane sterowania	D
24	DPOP	ALPHA	NL	Prognose	Besturingsgegevens	D
25	DPOP	ALPHA	RU	Прогноз	Управляющие данные	D
26	DPOP	ALPHA	ES	Pronostico	Datos de control	D
27	DPOP	ALPHA	SV	Prognos	Styrdata	D
28	DPOP	ANZPR	CS	Prognoza	Pocet obdobi povinny	D
29	DPOP	ANZPR	DE	Prognose	Anzahl der gewünschten Periode	D
30	DPOP	ANZPR	EN	Forecast	Number of periods required	D
31	DPOP	ANZPR	FR	Prevision	Nombre de periode requis	D
32	DPOP	ANZPR	IT	Previsione	Quantita del periodo richiesto	D
33	DPOP	ANZPR	DA	Prognose	Antal onskede perioder	D
34	DPOP	ANZPR	PL	Prognoza	Liczba okres obowiązkowy	D
35	DPOP	ANZPR	NL	Prognose	Aantal periode verplicht	D
36	DPOP	ANZPR	RU	Прогноз	число периоды обязательный	D
37	DPOP	ANZPR	ES	Pronostico	Cantidad dos periodos obligato	D
38	DPOP	ANZPR	SV	Prognos	Antal period obligatorisk	D
39	DPOP	BETA1	CS	Prognoza	Ridici data	D

Figure 2. Content of the single database table in the previous implementation

Every field in figure 1 is specified by the combination of the table name and field name (the first two columns, in figure 2). If, for example, we wanted to add a certain field in the previous database table, we had to carefully add lines in this table for every language, in order not to have typos, case in which the field was not displayed in the corresponding tab or area. Therefore, in order to avoid this, in the new implementation we want to keep track separately of tabs, areas and fields. Then, if we want to insert a new tab, this should be maintained in the tab page. In order to enter a new area, this area should be entered in the area page, corresponding to a certain tab. In order to add a specific field, this has to be added to a certain area, which corresponds to a certain tab.

2.2. Proposed solution based on cluster view

The proposed solution suggests the use of five database tables instead of the single table in figure 2. The five tables, named TAB, TABT, AREA, ARET and FLD, are combined within three views (TABV, AREV and FLDV) which build a cluster view, TAFC (figure 3).

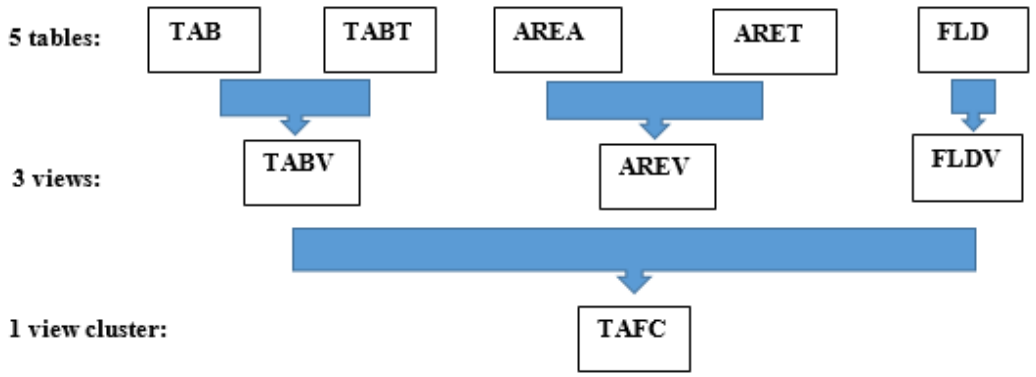

Figure 3. The logical data model of tables and views

The description of the objects in the figure above is specified in Table 1:

Table 1. Description of the objects used in the logical data model

Object Name	Description
Dictionary Objects	
Database Tables	
• TAB	Tabs for Popup
• TABT	Translation of Tabs
• AREA	Areas for Popup Tabs
• ARET	Translation for Areas
• FLD	Fields for Areas of Popup
Maintenance Views	
• TABV	View of Tabs and corresponding Translations
• AREV	View of Area for Tabs and corresponding Translations
• FLDV	View of Fields for Popup
View Cluster	
• TAFC	Table-Area-Field-Cluster

All database tables have the delivery class *C (customizing table)*, and *Display/Maintenance is allowed*. As for the technical settings, *APPL0 (Master data, transparent tables)* with the *size category 0* has been chosen.

The enhancement category for these database tables is: *Can be enhanced (character-type or numeric).*

The structure of the tables, primary keys and foreign keys are shown in figure 4:

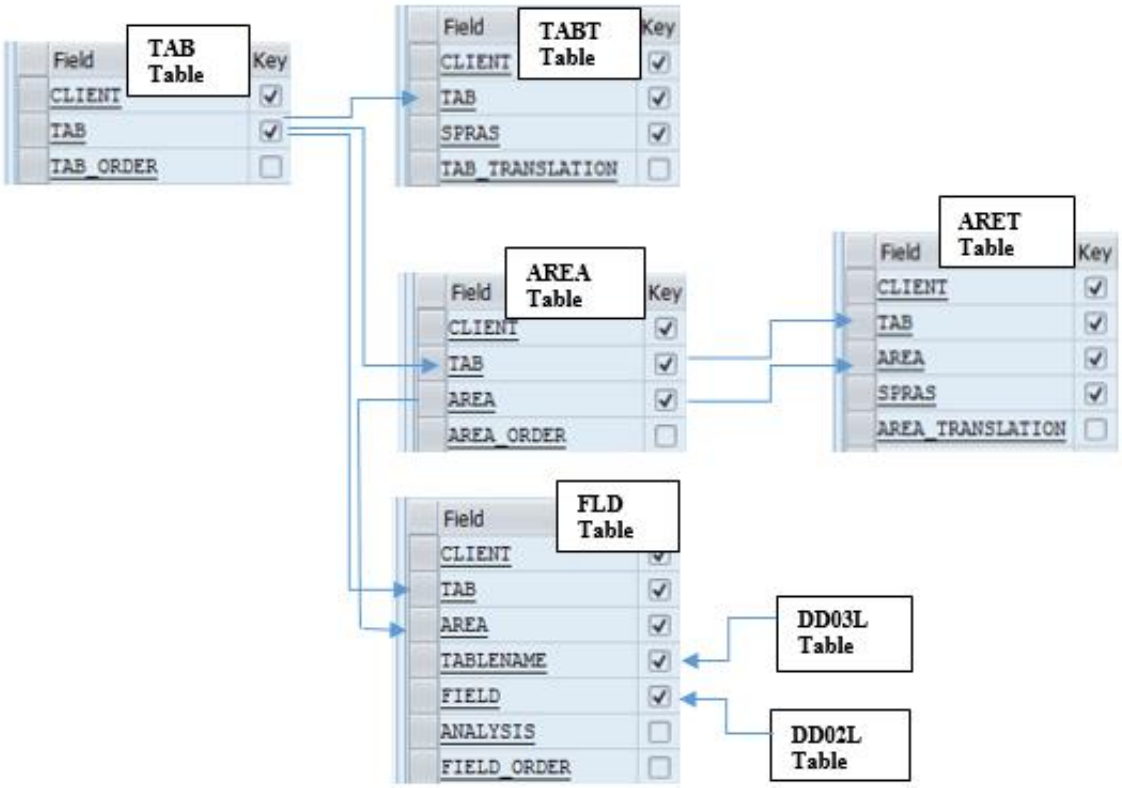

Figure 4. Table structure and relationships

The names of the fields in the database tables are relevant for their content. Only the SPRAS field in the translation-tables TABT and ARET must be explained: SPRAS is a system-field which stands for the language and is used in order to maintain the languages in which the tab/area is translated into.

We note that the main database tables (TAB, AREA and FLD) have a field called TAB_ORDER, AREA_ORDER or FIELD_ORDER, so the customer may arrange the tabs, areas and fields in the desired order.

The translation tables (TABT for the tabs and ARET for the areas) contain beside the data also a language field and a translation field for the corresponding language, which has a different data element assigned, in order to also display lower case characters.

Beside the tabs and the areas, the database table FLD also contains the fields TABLENAME and FIELD, which suggest the related parameters, whom input may be updated at runtime. Through standard SAP functionality the tables in BASIS, respectively their fields are by default translated in the login language of the user. So in the fields TABLENAME and FIELD of the FLD table, we will obtain, in the user login language, the names of the tables and fields from BASIS via the foreign keys to the table DD03L for TABLENAME and to the table DD02L for FIELD.

Beside the database tables, 3 maintenance views have been created, TABV, AREV and FLDV, for the tabs, areas and fields of the popup (figure 5). We have chosen maintenance views instead of database views to be able to use them in the view cluster.

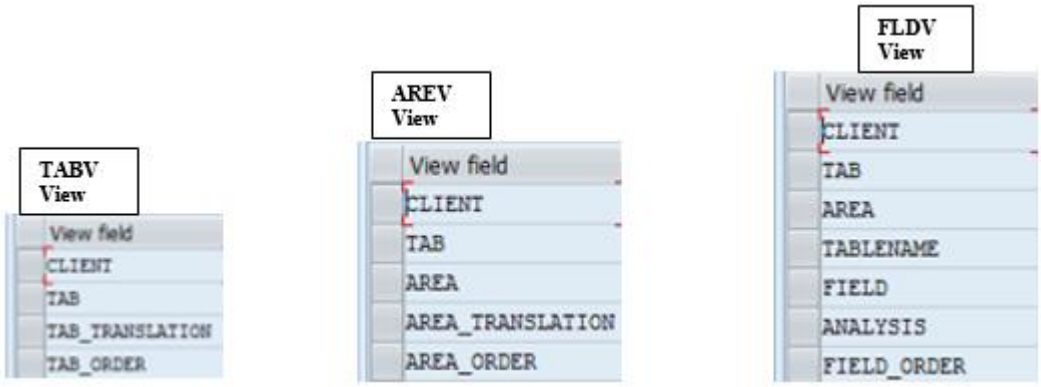

Figure 5. Maintenance Views

For all these tables and views, table maintenance generators have been created and activated, in order to have the possibility to manage individual datasets in every table and view. The corresponding names of those function groups for the table maintenance generators are the same names as those for the views.

The purpose of these maintenance views is only to take care of the input data more efficiently. These views will be used later in the view cluster, which ensures a hierarchical order of the data. Therefore, the maintenance views will also include the predecessor, in order to facilitate linking in the field dependency tab of the view cluster (Swapna, 2007).

These three maintenance views are embedded in the following view cluster (figure 6):

View cluster	TAFC			
Object structure				
View/Table	Short text	Predecessor	Dependency	Position
TABV	Tabs	TABV	R (Root)	1
AREV	Areas	TABV	S (Dependent on higher-level entry)	2
FLDV	Fields	AREV	S (Dependent on higher-level entry)	3

Figure 6. View cluster TAFC

Field dependency has been generated automatically and as a result the following desired structure has been achieved (figure 7):

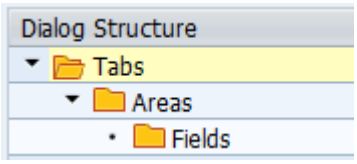

Figure 7. Dialog structure of the View Cluster

After populating the database tables with data in different languages with the help of the view cluster, the popup has been adapted in order to support the translation of the tabs and areas.

Supplementary internal tables have been defined in the function module POPUP_FLEX, in order to copy data from the translation tables. The corresponding SELECT statements have been embedded in TRY-CATCH blocks, in order to prevent short dumps due to faulty selection processes.

Similar to the previous implementation, the data regarding the tabs, areas and fields have been read in loops, corrected with avoiding SELECT statements within these loops. In case of embedding select statements within loops, this would affect the performance of the application, and would not be recommended by any programmer. Therefore, the select statements are situated right after the data declaration section and all data is selected from the database tables into internal tables. We can work further only using the internal tables at runtime, and avoid accessing the database within the loop statements every time we pass through that loop.

After selecting all the data we need for the loops and having all the necessary tabs, areas and fields in our internal tables, the corresponding translations have been accessed with the READ TABLE instruction separately, using the WITH KEY addition, so in case that one translation is missing, the name of the tab or area will be displayed instead.

In the INITIALIZATION event, the title bar and the GUI Status (Execute and Cancel buttons) have been adapted, in order to support language dependency.

The shared memory area classes have been adapted, similar to the previous implementation. Also, the text message classes remain the same.

Based on a new data organization and implying the introduction of three views and a view cluster the so-called language dependency has been improved. The redesigned application offers an easier maintainability of the data (fields, areas and tabs can be easily added, deleted, reordered and renamed) and all the data within the view cluster can be translated into any language supported by the system.

3. Advantages of the proposed solution

The proposed solution is a technical solution designed for the SAP MM module in order to improve language dependency according to the user logon language. The solution is recommended to be implemented in the SAP systems, the following benefits being ensured:

1. ***Consistent overview of existing tabs, areas and fields, as well as of the languages, in which a specific field of an area or tab is translated***

In the previous implementation, without having a unique name or code for a certain tab, but only the corresponding translations for different languages, there was no possibility to obtain (by sorting or filtering), the languages a certain tab was translated into.

Now, having a unique tab name, which is not dependent on any translation, we can view all existent translations with the help of the TAB and TAB_TRANSLATION fields of the TABT table. Similar, having a unique area name, which is not dependent on any translation, we can view all existent translations with the help of the AREA and AREA_TRANSLATION fields of the ARET table.

2. No previous translation search is necessary in order to add a field

In the previous implementation, we had to search for a translation, if we wanted to add a certain field of a specific area or tab. In case of not searching or finding the correct translation, just by adding directly a line, without verifying the correct predecessors, there was the possibility to add a new, but not identical translation. The newly added field could easily be interpreted as belonging to a new area or tab.

In the new implementation, using the view cluster, this can be avoided, by selecting a certain tab, which is now uniquely defined (not by the translation, but by the tab name, i.e. the TAB field in TAB table) and then by selecting the desired area, for adding a new field in the popup screen.

3. Easier maintainability of translations

As described earlier, the previous solution didn't provide any overview of the existing languages, in which the application was translated. One had to manually check the entries of the database table in order to add new data.

Using the view cluster solution, it's easy to select all tabs or only certain tabs, all or only certain areas, and translate them in one or more desired languages, using the Goto -> Translation option, provided by the main menu of the view cluster.

For example, we select in the view cluster the tab called "Forecasting" and then choose Goto -> Translation (figure 8). After selecting Goto -> Translation, we obtain a selecting window for the desired languages where the user can check one or more languages to translate those tabs or areas into.

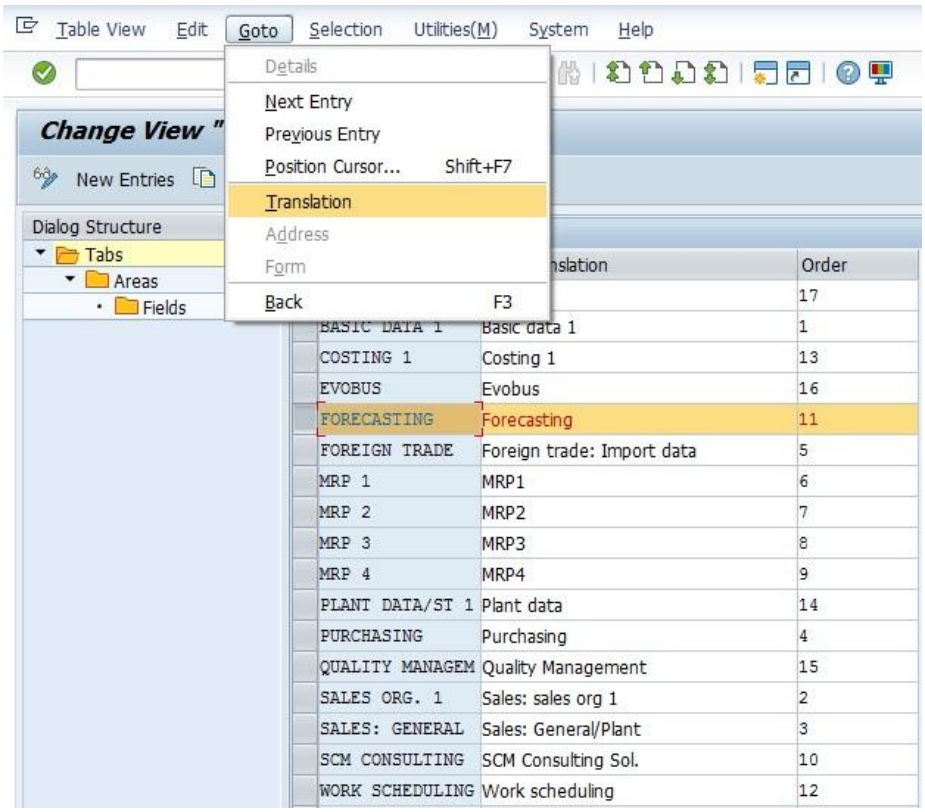

Figure 8. GoTo -> Translation Option

After checking a few (or all!) languages for instance and pressing the Ok button, we obtain the windows shown in figure 9a**,** or respectively 9b for all languages. Here we can add one or more missing translations, or modify one or more of the existing translations accordingly.

All the data within the view cluster can be translated into any language supported by the system.

Figure 9a. Translation of a tab for certain languages

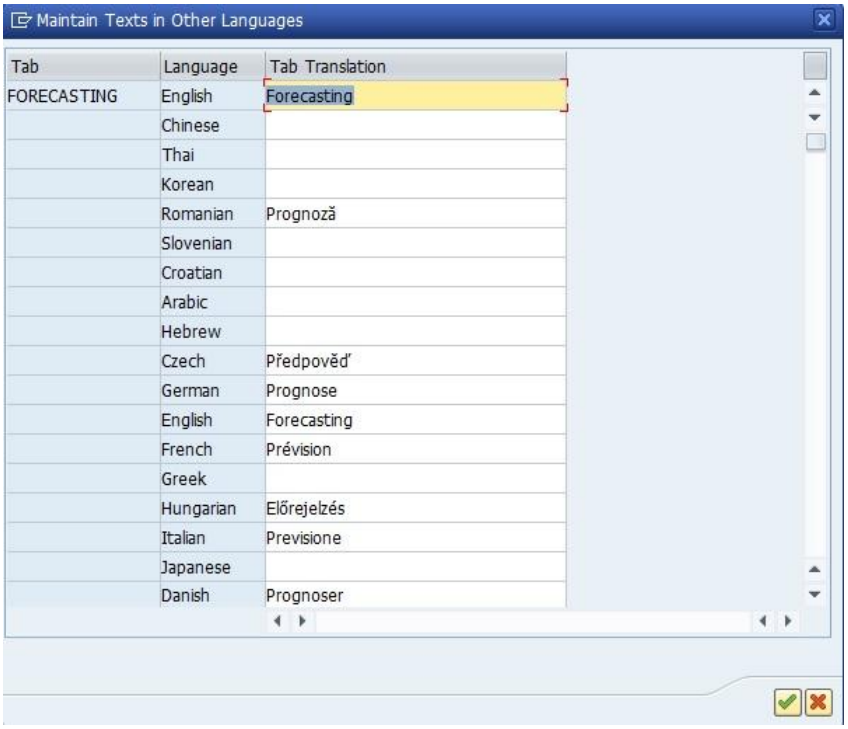

Figure 9b. Translation of a tab for all languages

4. Possibility to sort tabs, areas and fields based on an order number

In the previous implementation, there was no order defined for the tabs, areas and fields and therefore in the popup screen, the tabs, areas and fields appeared in the sequential order, as they were processed in the SQL SELECT command of the database table.

In the new implementation, in the TAB table is a TAB_ORDER field, in the AREA table is an AREA_ORDER field and in the FLD table is a FIELD_ORDER field, where the user can define the desired order for the corresponding item.

5. Avoiding human error regarding missing translations

In the previous solution, all data was selected into an internal table at runtime, depending on the user login language. If no data was found in the login language of the user, the default language was set to English. Further, the tabs and areas have been chosen from the internal table using the SELECT DISTINCT command. This approach lacks the ability to display an area, which has not been translated. Besides, there hasn't been implemented a solution in case of a void result of the SELECT statement.

The new implementation corrects this behavior by using the TRY-CATCH blocks. If the result of a selection is void, the block catches the error and raises an exception inside the function module. At runtime, the new implementation provides the selection of the fields from the FLD database table. Within the SELECT statement has been used the ORDER BY clause. For the next

TRY-CATCH block, the SELECT statement has been adapted with the FOR ALL ENTRIES IN clause, in order to select data, which has a correspondent in the internal table of the fields. This clause is Open SQL specific and is often used in ABAP programs (Hardy, 2015). The only disadvantage of using this clause will be the fact that we can no longer use the ORDER BY clause. The solution for this matter is easy to solve by using an ABAP SORT command after the TRY-CATCH block. After the selection of the tabs and areas, the corresponding translation is selected into other internal tables at runtime.

4. Conclusions

Material masterdata is an important area, and therefore the corresponding translations play a key role. The most important group in this category is the material and product data maintenance. Because a global enterprise offers identical products in many countries, the product description fields must be translated into all the languages of these countries.

In this paper we present a way to modify the underlying schema of the original SAP Material Master data Application in order to facilitate the management of tabs, areas and fields contained in the popup layout, for different languages. The previous implementation of the language dependency was based on a single database table that couldn't provide a consistent overview of the existing tabs, areas and fields. Further, there was no possibility to select the appearances of a specific field in all translations. Improvements have been required in order to increase the flexibility of the function module. Our proposal is based on a cluster view implementation described in detail in paragraph 2.2. The benefits of this approach are evident in terms of consistency and ease of maintenance. Disadvantages like updating manually the database table for inserting, deleting or renaming tabs, areas and/or fields, no possibility to change the sequential order for tabs, areas and fields in the popup screen, or no possibility to view all translations for a certain tab or area, have been eliminated. Furthermore, during the testing phase, the following advantages have been validated: 1- Consistent overview of data; 2- No preliminary preparation is necessary for adding data; 3- Easier maintainability of translations; 4- Possibility to sort data based on user choice; 5- Bypassing human negligence.

The proposed solution is dedicated for SAP systems, concrete technical aspects of the SAP MM module being taken into consideration. However, the approach framework can be applied to similar demarches of other software applications.

References

Johnson, M. (2013). SAP Material Master: A Practical Guide, Expresso Tutorials GmbH

Ashfaque, A. (2014). The SAP Material Management Handbook, CRC Press - Taylor & Francis Group, LLC

Jacob, R. (2012). Language Installation on SAP systems - Steps by steps. Retrieved from http://scn.sap.com/community/netweaver-administrator/blog/2012/12/10/installation-of-languages-on-sap-systems

Ong, O. (2014). SAP Language Handling and Language Installation Procedure, Retrieved from https://www.academia.edu/15795712/SAP_Language_Handling_and_Language_Installation_Procedure

Hardy, P. (2015). ABAP to the Future, Rheinwerk Publishing Boston

Swapna, T.N. (2007). Creating a View Cluster, Retrieved from https://www.academia.edu/15795712/SAP_Language_Handling_and_Language_Installation_Procedure