Best Practices in SAP Environment

MIHAELA MUNTEAN, DIANA TÂRNĂVEANU
Business Information Systems Department
West University of Timişoara
Str. V. Pârvan, nr. 4, Timişoara
ROMANIA
mihaela.muntean@e-uvt-ro, diana.tarnaveanu@e-uvt.ro, http://www.feaa.uvt.ro

Abstract: - Today's economy is strongly influenced by information technology, therefore organizations have to deal with many challenges like integrated information and consistency, security, advanced reporting and customer service facilities. Finding relevant information to base their decision on, is one of the greatest challenge managers have to face. SAP is a highly flexible, customized solution, where the information is processed once and is accessible to all modules. It offers many reporting services, the goal for the developers being using the most efficient tool to find the solution of the problem. This paper presents best practices in SAP Environment through two study cases, using reporting tools specific to SAP ABAP and SAP Business Warehouse module.

Key-Words: - decision making, best practices, SAP ABAP, SAP Business Warehouse

1 Retrieving relevant information and decision making

Many organizations often implement an enterprise resource planning solution, like SAP; all the individual department functions being integrated into a single software application [1]. Out of many advantages, improved productivity, increase efficiency, decrease costs and streamline processes stand out [2]. SAP makes it easier to track the workflow across various departments. Alongside other reporting services, it can also integrate Business Intelligence functionalities that can give overall insights on business processes and identify potential areas of problems/improvements.

Managers need support in every step of the decision process; a tailored solution should be created for manipulating data, searching the relevant information and performing complex calculus, time being an important resource.

2 Queries using SAP ABAP

ABAP (Advanced Business Application Programming – originally Allgemeiner Berichts-Aufbereitungs-Prozessor, German for „general report creation processor”) is a programming language developed by SAP for programming business applications in the SAP environment [3] [4]. ABAP is an interpretative 4GL which supports structured programming and modularization; it has been enhanced as an object-oriented language; it is capable of handling multi-language applications; fully integrates an SQL standard and is platform-independent [5].

2.1 Basic ABAP concepts

The most common ABAP object is the report. Reports are programs which generate lists of data. They may involve a small amount of interactivity, but mainly they supply data to the front-end interfaces, the SAP GUI, etc. When a user runs a report, typically at the first step a selection screen is displayed. Once the selection parameters are inserted and the report is executed, they cannot intervene in the execution of the program. The program runs and then displays the output.

Modularization means taking sequence of ABAP statements and placing them in their own, separate module. Those modules can be called from the program. Modularization allows single tasks to be focused upon one at a time, without the distraction and confusion which can be caused if the code is in the middle of a large, complicated structure. Creating individual source code modules prevents from having to write the same statements again and again. It is also of a great help in the case of support [6].

Two techniques can be used for local program modularization in the ABAP programming language: subroutines (also known as FORM routines) and methods in local classes. Even if it is technically possible to call a subroutine from another program, it is not advisable because it contradicts the principle of encapsulation of data and functions. There are also two techniques for global modularization in the ABAP programming language: function modules that are organized into function groups and methods in global classes. Globally defined modularization units can be used by any number of programs in the same time. They are stored centrally in the Repository and loaded when required [7]. All global variables defined in the main program can be addressed from a subroutine. But in order to formalize the subroutine and make it work with different data objects for each situation, the global variables are replaced with placeholders called formal parameters, and together form the subroutine interface. Whenever a subroutine is define, the interface must be declared. When the subroutine is called, formal parameters must be specialized by means of corresponding global variables (actual parameters), in order to reference the subroutine processing to real variables. This assignment of actual parameters to formal parameters when calling a subroutine is called parameter passing [8].

Function Groups are a special type of ABAP program; they are not executable programs, so they can’t be called using a transaction code. Their only purpose is to serve as main programs for function modules. Generally, a Function Group contains many functions modules which perform related functions or that operate on similar data. Function Modules are part of programs that can be stored globally, and in this way used by all ABAP programs. Function modules perform tasks of general interest to other programmers [9]. Usually these tasks are well-defined functions that all users need, regardless of application. Some well-defined tasks include performing tax calculations, determining factory calendar dates, and calling frequently-used dialogs. By using them, the developer can take advantage of modularization [10].

ABAP programs use screens to obtain input from users. The most general type of screen is a dialog screen, which you create using the ABAP Workbench tools Screen Painter and Menu Painter. These tools allow to create screens for data input and output. You often use screens purely for data input. In these cases, you can use a selection screen. Selection screens provide a standardized user interface in the R/3 System. Users can enter both single values and complex selections. Input parameters are primarily used to control the program flow, while users can enter selection criteria to restrict the amount of data read from the database [110]. ALV stands for ABAP List Viewer, as named initially, because it was only available in ABAP. It is now a more general concept, which is available in Java too, and GRID was introduced later. ALV is a generic tool that outputs data in a table form (rows and columns) with integrate functions to manipulate output and export it. It is also possible to make ALV editable via ALV control. SAP provides a set of ALV function modules which can be put into use to embellish the output of a report. ALV provides inbuilt functions to the reports such as: sorting of records, filtering of records, totals and sub-totals, download the report output to Excel/HMTL, changing the order of the columns in the report and hide the unwanted columns from the report [12]. The ABAP Dictionary manages the database tables and contains current information about a database table's technical attributes [9]. It is the central facility in the SAP system where tables and other objects are created and maintained [13].

A transparent table in the dictionary has a one-to-one relationship with a table in the database. Its structure in R/3 Data Dictionary corresponds to a single database table. For each transparent table definition in the dictionary, there is one associated table in the database [14]. The database table has the same name, the same number of fields, and the fields have the same names as the R/3 table definition. They are used to hold application data. Application data is the master data or transaction data used by an application. We can either use native SQL or OPEN SQL to access transparent tables. Secondary indexes can be created and buffering can be done in transparent tables [15].

2.2 Study Case – Querying a Sales Order

Information about a sales/order document are stored in VBAK table, the details of the order in VBAP, the other related tables maintaining information regarding sales requirements, release order data, delivery due, schedule lines, shipping, billing, etc – Fig. 1.

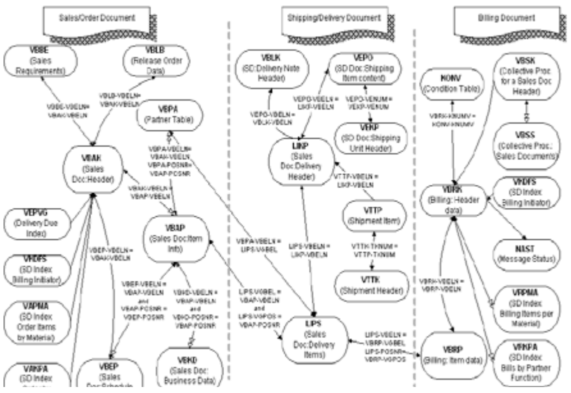
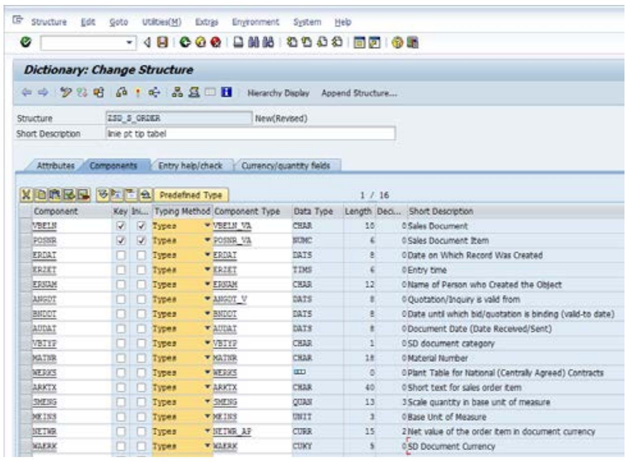

Sales/Order Document
Shipping/Delivery Document
Billing Document
VBBE (Sales Requirements)
VBLB (Release Order Data)
VBPA (Partner Table)
VBAK (Sales Doc:Header)
VBAP (Sales Doc:Item Info)
VBEP (Sales Doc:Schedule
VBKD (Sales Doc: Business Data)
VEPVG (Delivery Due Index)
VKDFS (SD Index Billing Initiator)
VAPMA (SD Index Order Items by Material)
VAKPA (SD Index
VBUK (SD:Delivery Note Header)
VEPO (SD Doc:Shipping Item content)
VEKP (SD Doc:Shipping Unit Header)
LIKP (Sales Doc:Delivery Header)
VTTP (Shipment Item)
VTTK (Shipment Header)
LIPS (Sales Doc:Delivery Items)
KONV (Condition Table)
VBSK (Collective Proc for a Sales Doc Header)
VBSS (Collective Proc: Sales Documents)
VBRK (Billing: Header data)
VKDFS (SD Index Billing Initiator)
NAST (Message Status)
VRPMA (SD Index Billing Items per Material)
VBRP (Billing: Item data)
VRKPA (SD Index Bills by Partner Function)

Fig. 1. Sales and Distribution Tables [16]

This study case offers an example of a modularized ABAP program used for querying the tables and displaying relevant information in a list, offering the manager relevant information in due time.

2.2.1 Output table

The retrieved relevant information should be saved in a transparent table created based on a global table type. The global table type is built based on a structure which should be created having the desired output fields with the same types and technical attributes as the fields from the inquired tables – Fig.2.



Dictionary: Change Structure

Structure ZSD_S_ORDER New(Revised)
Short Description Inie pt tip tabel

1 / 16

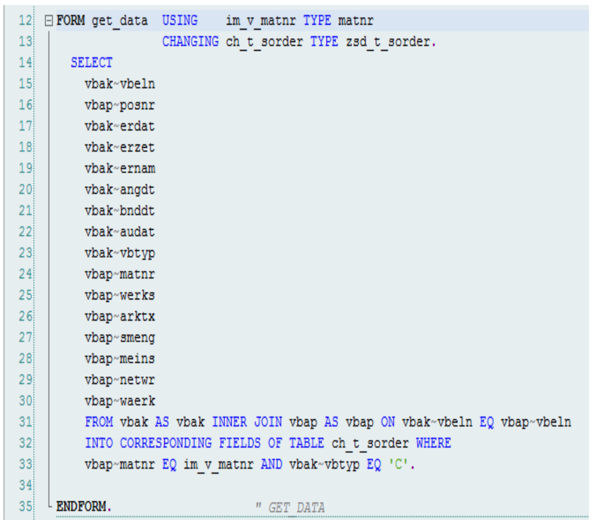
Component	Key	Ini...	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
VBELN	✓	✓	Types	VBELN_VA	CHAR	10	0	Sales Document
POSNR	✓	✓	Types	POSNR_VA	NUMC	6	0	Sales Document Item
ERDAT	☐	☐	Types	ERDAT	DATS	8	0	Date on Which Record Was Created
ERZET	☐	☐	Types	ERZET	TIMS	6	0	Entry time
ERNAM	☐	☐	Types	ERNAM	CHAR	12	0	Name of Person who Created the Object
ANGDT	☐	☐	Types	ANGDT_V	DATS	8	0	Quotation/Inquiry is valid from
BNDDT	☐	☐	Types	BNDDT	DATS	8	0	Date until which bid/quotation is binding (valid-to date)
AUDAT	☐	☐	Types	AUDAT	DATS	8	0	Document Date (Date Received/Sent)
VBTYP	☐	☐	Types	VBTYP	CHAR	1	0	SD document category
MATNR	☐	☐	Types	MATNR	CHAR	18	0	Material Number
WERKS	☐	☐	Types	WERKS		0	0	Plant Table for National (Centrally Agreed) Contracts
ARKTX	☐	☐	Types	ARKTX	CHAR	40	0	Short text for sales order item
SMENG	☐	☐	Types	SMENG	QUAN	13	3	Scale quantity in base unit of measure
MEINS	☐	☐	Types	MEINS	UNIT	3	0	Base Unit of Measure
NETWR	☐	☐	Types	NETWR_AP	CURR	15	2	Net value of the order item in document currency
WAERK	☐	☐	Types	WAERK	CUKY	5	0	SD Document Currency

Fig. 2. Structure's fields

The selected fields are: sales document (vbeln), sales document item (posnr), created on (erdat), time (erzet), created by (ernam), quotation valid from (angdt), quotation valid to (bnddt), document date (audat), sd document category (vbtyp), material (matnr), plant (werks), etc. Special attention require the fields netwr (net values) and waerk (document currency) – Fig. 3. They should be built based on similar fields with the same respective types.

2.2.2 Querying the SD tables

A function group should be created, as a container for the function module. The function module's objective is to query the Sales and Distribution tables: vbak and vbap. The import parameter is the material number (matnr), the export parameter is the output table. Source Code tab will contain the code that queries the tables. The function module will be saved in a function group, and can be called from any other report, making the code more efficient. The query is displayed in Fig. 3.



```
FORM get_data  USING    im_v_matnr TYPE matnr
               CHANGING ch_t_sorder TYPE zsd_t_sorder.
  SELECT
    vbak~vbeln
    vbap~posnr
    vbak~erdat
    vbak~erzet
    vbak~ernam
    vbak~angdt
    vbak~bnddt
    vbak~audat
    vbak~vbtyp
    vbap~matnr
    vbap~werks
    vbap~arktx
    vbap~smeng
    vbap~meins
    vbap~netwr
    vbap~waerk
    FROM vbak AS vbak INNER JOIN vbap AS vbap ON vbak~vbeln EQ vbap~vbeln
    INTO CORRESPONDING FIELDS OF TABLE ch_t_sorder WHERE
    vbap~matnr EQ im_v_matnr AND vbak~vbtyp EQ 'C'.

ENDFORM.                    " GET_DATA
```

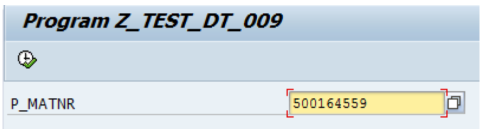
Fig. 3. The query

2.2.3 Creating the screen and calling the function

When creating the screen, a parameter for material number is needed. It will have the same data type as the one from the vbap table.

SELECTION-SCREEN BEGIN OF BLOCK 1.
   PARAMETERS: p_matnr TYPE vbap-matnr.
SELECTION-SCREEN END OF BLOCK 1.

On the execution screen, when the input parameter is inserted, the query is performed with its value – Fig. 4.

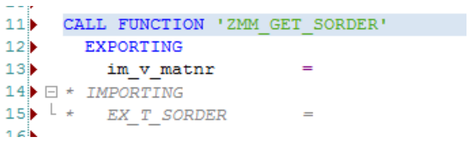


Program Z_TEST_DT_009

P_MATNR 500164559

Fig. 4. Material number – Input parameter

In order to call the function, the Pattern button should be used, facilitating the creation of the function. The function can then be easily customized, some clauses can be dropped and the other just uncommented and ready to be used – Fig. 5.



```
CALL FUNCTION 'ZMM_GET_SORDER'
  EXPORTING
    im_v_matnr        =
* IMPORTING
*   EX_T_SORDER       =
```

Fig. 5. Inserting a function using Pattern button

When customizing the function, actual importing and exporting parameters are used.

```
FORM read_data USING im_v_matnr1 type vbap-
matnr CHANGING ch_t_vbap1 type zsd_t_sorder.
CALL FUNCTION 'ZMM_GET_SORDER'
  EXPORTING
    im_v_matnr  = im_v_matnr1
  IMPORTING
    ex_t_sorder = ch_t_vbap1.
ENDFORM.
```

2.2.4 Displaying the result

For displaying the result the ALV functionality is used. REUSE_ALV_GRID_DISPLAY is a standard SAP function module that performs output of a simple list (single-line) functionality. Its pattern include import, export, tables parameters and a list of exceptions. A field catalog is prepared using the internal table (LT_FIELDCAT) of type SLIS_T_FIELDCAT_ALV.[xx]

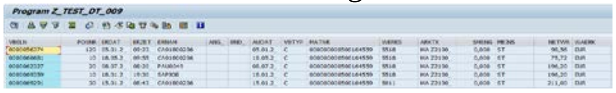
```
FORM alv_display USING im_t_vbap TYPE zsd_t
_sorder.
  TYPE-POOLS: slis.
  DATA: lt_fieldcat TYPE slis_t_fieldcat_alv,
        ls_fieldcat TYPE slis_fieldcat_alv,
        ls_layout TYPE slis_layout_alv.

  PERFORM alv_fieldcat CHANGING lt_fieldcat.
  CALL FUNCTION 'REUSE_ALV_GRID_DISPL
AY'
    EXPORTING
      i_callback_program = sy-repid
      is_layout          = ls_layout
      it_fieldcat        = lt_fieldcat
    TABLES
      t_outtab           = im_t_vbap
    EXCEPTIONS
      program_error      = 1
      OTHERS             = 2.
  IF sy-subrc <> 0.
    MESSAGE e001(00) WITH 'Atentie, eroare
ALV!'.
  ENDIF.
ENDFORM.
```

Field catalog contains descriptions of the list output fields (usually a subset of the internal output table fields). The field catalog for the output table is built-up in the caller's coding [x].

```
FORM alv_fieldcat CHANGING ch_t_fieldcat TYP
E slis_t_fieldcat_alv.
  DATA: ls_fieldcat TYPE slis_fieldcat_alv.
  ls_fieldcat-fieldname = 'VBELN'.
  ls_fieldcat-key = 'X'.
  ls_fieldcat-reptext_ddic   = 'VBELN'(003).
  APPEND ls_fieldcat TO ch_t_fieldcat.
  CLEAR: ls_fieldcat.
…
  ls_fieldcat-fieldname = 'WAERK'.
  ls_fieldcat-reptext_ddic = 'WAERK'.
  APPEND ls_fieldcat TO ch_t_fieldcat.
  CLEAR: ls_fieldcat.
END FOR.
```

The results are shown in Fig. 6.



Program Z_TEST_DT_009

VBELN	POSNR	ERDAT	ERZET	ERNAM	ABG..	ABD..	AUDAT	VBTYP	MATNR	WERKS	ARKTX	SMENG	MEINS	NETWR	WAERK
[illegible]	120	[illegible]	[illegible]	[illegible]			[illegible]	C	[illegible]	[illegible]	[illegible]	0,000	ST	[illegible]	EUR
[illegible]	10	[illegible]	[illegible]	[illegible]			[illegible]	C	[illegible]	[illegible]	[illegible]	0,000	ST	79,72	EUR
[illegible]	30	[illegible]	[illegible]	[illegible]			[illegible]	C	[illegible]	[illegible]	[illegible]	0,000	ST	196,20	EUR
[illegible]	10	[illegible]	[illegible]	SAP90B			[illegible]	C	[illegible]	[illegible]	[illegible]	0,000	ST	196,20	EUR
[illegible]	30	[illegible]	[illegible]	[illegible]			[illegible]	C	[illegible]	[illegible]	[illegible]	0,000	ST	211,80	EUR

Fig. 6. The queried tables

This example shows a useful and time-efficient way to query the tables and obtain relevant information for an order sale.

3 SAP BW Infoproviders for reporting

3.1 Introducing the concepts

Reporting in the SAP BW environment implies the access to and the process of multiple data sources in a single report [20]. Reports are query-based objects, designed to deliver relevant information to the end-users and enable business analysis. In SAP BW, the InfoProviders are objects that provide data for a query. They can be persistent, also called data targets, like InfoObject, DataStoreObject (DSO) and InfoCube or non-persistent, like MultiProvider, InfoSet and VirtualProvider [20].

Working with queries in the SAP BW environment is limited by the fact that a query can only be based on a single InfoProvider. Therefore, the non-persistent InfoProviders are used to combine data from various persistent

InfoProviders and present the result as if they were one source [20].

InfoObjects are the basis for defining or configuring all of the other InfoProviders. They are divided into characteristics (for example, customers), key figures (for example, revenue), units (for example, currency, amount unit), time characteristics (for example, fiscal year) and technical characteristics (for example, request number). Characteristics are sorting keys, such as company code, product, customer group, fiscal year, period, or region. They specify classification options for the dataset and are therefore reference objects for the key figures.

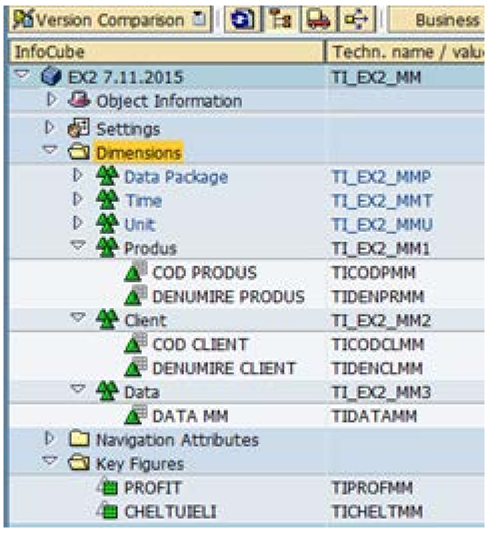
In the InfoCube the characteristics are stored in dimensions. The key figures provide the values that are reported on in a query. Key figures can be quantity, amount, or number of items. They form the data part of an InfoProvider. InfoCubes are mainly used for multidimensional reporting (OLAP) and are the primary objects used to support queries. They physically store data and are optimized for the performance of queries [20]. An InfoCube is design as a star schema. The fact table contains the key figures and the dimension IDs as foreign keys. SAP BW uses an extended star schema for InfoCubes. The schema is extended because the master data used to build the dimension tables is replaced by keys again [20].
A DSO stores data in transparent tables. Data is extracted and unified at a very detailed level, and for that reason, DSOs are not optimized for reporting purpose. Key fields and data fields are the two types of DSO components. Key fields are InfoObjects that uniquely identify each line. Data fields contain characteristics and key figures loaded from the operational system. Although it is not recommended to create queries from DSOs, it can be done if required.

MultiProviders combine data of several InfoProviders and provide a single view of the data. That means that a union operation takes place at the database level, combining all values for the InfoProviders. MultiProviders do not store data: a query collects data directly from the InfoProviders. To improve the performance of a query, a MultiProvider should preferably be based only on InfoCubes.
InfoSets do not store data: they combine data from several InfoProviders. They use a join operation to display data instead of the union operation used by the MultiProviders. Four types of join operations are possible in InfoSets: inner join, left outer join, temporal join and anti-join [20].

VirtualProviders access data in real-time directly from a source system. They function like InfoCubes, except that they do not store any data in the SAP BW system. There are three options for creating a VirtualProvider: based on the data transfer process for direct access; based on a BAPI; based on a function module. They should be used for queries with small datasets and for few users because the system has to execute the entire process in real time [20].

3.2 Study case for Profit Analysis

In order to sustain a profit analysis by analytics, a Profit InfoCube is recommended to be designed (Figure 7). Starting with some elementary characteristics and two key figures, the profit analysis will be conducted. In order to create the Data Source, the corresponding structure will be defined with respect to the Source structure (the Source contains information stored in an Excel sheet). The proposed cube has three dimensions: Produs, Client and Data and two Key Figures (Measures).



InfoCube	Techn. name / valu
EX2 7.11.2015	T1_EX2_MM
Object Information	
Settings	
Dimensions	
Data Package	T1_EX2_MMP
Time	T1_EX2_MMT
Unit	T1_EX2_MMU
Produs	T1_EX2_MM1
COD PRODUS	T1CODPMM
DENUMIRE PRODUS	T1DENPRMM
Client	T1_EX2_MM2
COD CLIENT	T1CODCLMM
DENUMIRE CLIENT	T1DENCLMM
Data	T1_EX2_MM3
DATA MM	T1DATAMM
Navigation Attributes	
Key Figures	
PROFIT	T1PROFMM
CHELTUIELI	T1CHELTMM

Fig. 7. Profit InfoCube [19]

Dimension and Key Figure of the InfoCube are associated with elements of the Data Source (Figure 8), this process being identified by Transformation.

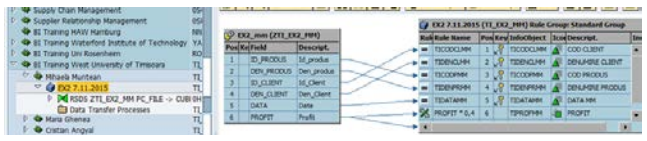


Fig. 8. Create Transformation [19]

For activating the Extract-Transform-Load process from the Source System into the InfoCube an InfoPackage is needed followed by the execution of the Data Transfer Process (DTP) [20]. The Infocube by itself can be an InfoProvider for reporting or as part of a MultiProvider together with other InfoProviders can support advanced reporting.

4 Reporting for Decision Making

4.1 Reporting with BEx

SAP BW is the Business Intelligence solution provided by SAP for reporting and data analysis. Data is extracted and loaded into SAP BW after being identified from different systems. It is than transformed into multi-dimensional structures to prepare it for analyses.

SAP BW has three architecture layers: 1-The data acquisition layer; 2-The data warehouse layer; 3-The reporting layer (Figure 9). The Business Explorer (BEx - Figure 10, 11) component provides users with extensive analysis options.

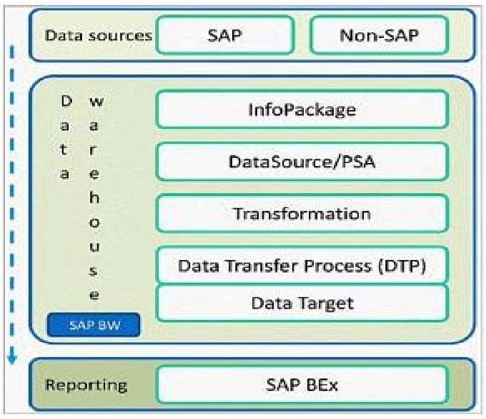


Fig. 9. SAP BW architecture [20]

The data is collected from SAP and non-SAP source systems and is extracted in pull mode using InfoPackage objects. It is then temporally store in the staging area (PSA) before the transformations from the source format to the desired destination format take place. Afterwards, the loading process of adding transformed data to data targets using the Data Transfer Process (DTP) begins. The reporting layer finally presents the data in reports (Figure 12) and dashboards useful for decision-making.

Back to the study case, in the BEx Query Designer the report will be developed (Figure 10).

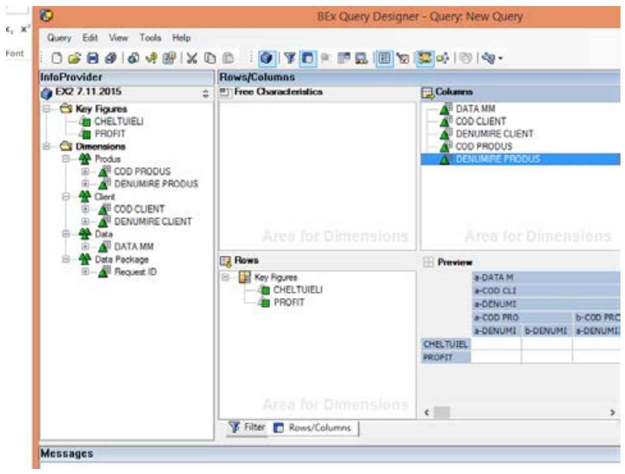


Fig. 10. BEx Query Designer

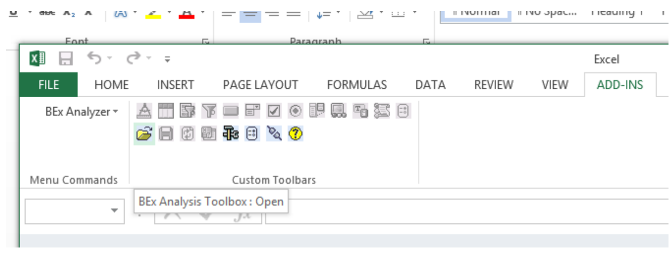


Fig. 11. BEx Analyzer

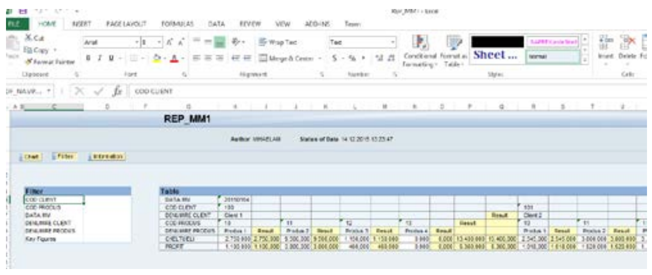


Fig. 12. Displaying the data

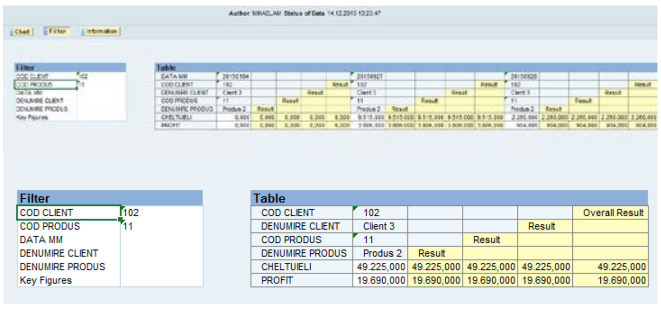
Based on the end-user requirements, the following derivated reports have been generated.

Request 1. PROFIT and CHELTUIELI for COD PRODUS = 11 and COD CLIENT 102 in detailed and a synthetic form (Figure 13).

Request 2. Display the daily Total_PROFIT and Total_CHELTUIELI (Figure 14).

Request 3. Display PROFIT and CHELTUIELI in a

CHART diagram by illustrating the impact of each product (Figure 15).



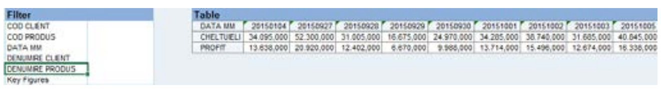
Filter	
COD CLIENT	102
COD PRODUS	11
DATA MM	
DENUMIRE CLIENT	
DENUMIRE PRODUS	
Key Figures	

Table															
DATA MM	20150104					20150927					20150928				
COD CLIENT	102				Result	102				Result	102				Result
DENUMIRE CLIENT	Client 3			Result		Client 3			Result		Client 3			Result	
COD PRODUS	11		Result			11		Result			11		Result		
DENUMIRE PRODUS	Produs 2	Result				Produs 2	Result				Produs 2	Result			
CHELTUIELI	0,000	0,000	0,000	0,000	0,000	9.515,000	9.515,000	9.515,000	9.515,000	9.515,000	2.260,000	2.260,000	2.260,000	2.260,000	2.260,000
PROFIT	0,000	0,000	0,000	0,000	0,000	3.806,000	3.806,000	3.806,000	3.806,000	3.806,000	904,000	904,000	904,000	904,000	904,000

Filter	
COD CLIENT	102
COD PRODUS	11
DATA MM	
DENUMIRE CLIENT	
DENUMIRE PRODUS	
Key Figures	

Table					
COD CLIENT	102				Overall Result
DENUMIRE CLIENT	Client 3			Result	
COD PRODUS	11		Result		
DENUMIRE PRODUS	Produs 2	Result			
CHELTUIELI	49.225,000	49.225,000	49.225,000	49.225,000	49.225,000
PROFIT	19.690,000	19.690,000	19.690,000	19.690,000	19.690,000

Fig. 13. Request1. Displaying the result



Filter
COD CLIENT
COD PRODUS
DATA MM
DENUMIRE CLIENT
DENUMIRE PRODUS
Key Figures

Table									
DATA MM	20150104	20150927	20150928	20150929	20150930	20151001	20151002	20151003	20151005
CHELTUIELI	34.095,000	52.300,000	31.005,000	16.675,000	24.970,000	34.285,000	38.740,000	31.685,000	40.845,000
PROFIT	13.638,000	20.920,000	12.402,000	6.670,000	9.988,000	13.714,000	15.496,000	12.674,000	16.338,000

Fig. 14. Request 2. Displaying the data

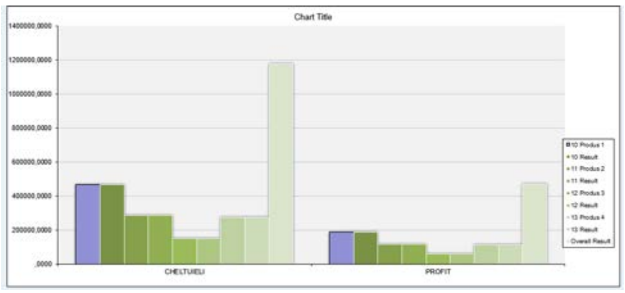
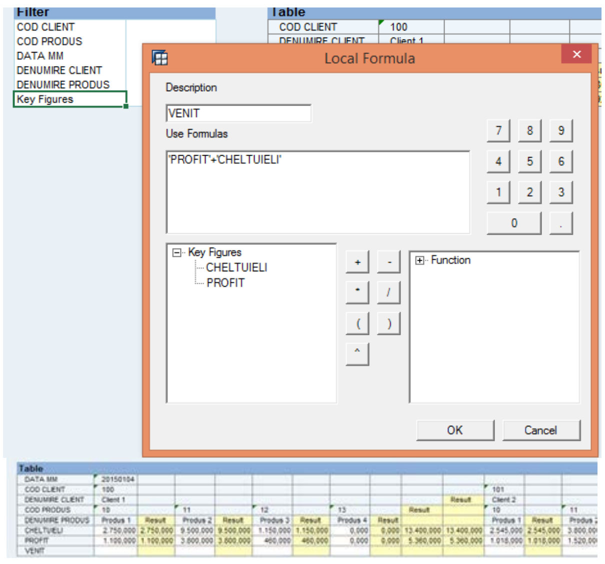


Fig. 15. Request 3. Displaying the data

Request 4. Introduce VENIT in the report (Figure 16, 17).



Description: VENIT

Use Formulas: 'PROFIT'+'CHELTUIELI'

Key Figures
- CHELTUIELI
- PROFIT

Table													
DATA MM	20150104												
COD CLIENT	100										101		
DENUMIRE CLIENT	Client 1									Result	Client 2		
COD PRODUS	10		11		12		13		Result		10		11
DENUMIRE PRODUS	Produs 1	Result	Produs 2	Result	Produs 3	Result	Produs 4	Result			Produs 1	Result	Produs 2
CHELTUIELI	2.750,000	2.750,000	9.500,000	9.500,000	1.150,000	1.150,000	0,000	0,000	13.400,000	13.400,000	2.545,000	2.545,000	3.800,000
PROFIT	1.100,000	1.100,000	3.800,000	3.800,000	460,000	460,000	0,000	0,000	5.360,000	5.360,000	1.018,000	1.018,000	1.520,000
VENIT													

Fig. 16. Adding a new, calculated field in the report

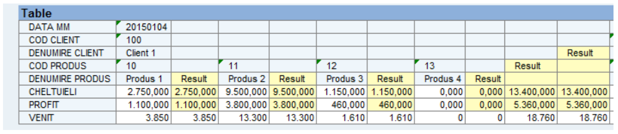


Table										
DATA MM	20150104									
COD CLIENT	100									
DENUMIRE CLIENT	Client 1									Result
COD PRODUS	10		11		12		13		Result	
DENUMIRE PRODUS	Produs 1	Result	Produs 2	Result	Produs 3	Result	Produs 4	Result		
CHELTUIELI	2.750,000	2.750,000	9.500,000	9.500,000	1.150,000	1.150,000	0,000	0,000	13.400,000	13.400,000
PROFIT	1.100,000	1.100,000	3.800,000	3.800,000	460,000	460,000	0,000	0,000	5.360,000	5.360,000
VENIT	3.850	3.850	13.300	13.300	1.610	1.610	0	0	18.760	18.760

Fig. 17. Displaying the data

This study case illustrate how SAP BW can make decision making more effective, by offering information from many different data sources and displaying tailored reports based on the manager's requests.

5 Conclusion

The cost of a wrong decision making can be high, the effect being direct or indirect, like a chain reaction, on other elements of the organization. The quantity of the information that is constantly growing and market fluctuations makes decision making difficult. Information technologies can assist managers in choosing from the multitude of alternatives.

Considering that SAP is implemented in the organization, we proposed some best practices materialized in two study cases in SAP ABAP and SAP BW that support the manager in taking a time and cost-efficient decision.

References:

[1] X1.http://www.excitingip.com/2010/advantages-disadvantages-of-erp-enterprise-resource-planning-systems/

[2] X2.http://www.workwisellc.com/5-benefits-implementing-erp-software/

[3] X3.http://searchsap.techtarget.com/definition/ABAP

[4] X4.Horst Keller, Sascha Kruger, *ABAP Objects. ABAP Programming in SAP NetWeaver*, SAP Press, Galileo Press, 2011

[5] X5.https://en.wikipedia.org/wiki/ABAP#cite_note-1

[6] X6.https://help.sap.com/saphelp_nw70/helpdata/en/9f/db970e35c111d1829f0000e829fbfe/content.htm

[7] X7.http://www.saphub.com/abap-tutorial/modularization-in-abap/

[8] X8.http://it.toolbox.com/wiki/index.php/What_are_Modularization_Techniques%3F

[9] X9. Peter Moxon, *Beginner's guide to SAP ABAP. An introduction to programming SAP*

applications using ABAP, Sapprouk Limited, 2012

[10] X10.https://www.sapnuts.com/courses/core-abap/modularization-in-abap/create-function-modules.html

[11] X11. http://www.sapdev.co.uk/dialog/basic-dialog.htm

[12] X12.https://wiki.scn.sap.com/wiki/display/ABAP/ALV

[13] X13.https://help.sap.com/saphelp_nw73ehp1/helpdata/en/cf/21ea0b446011d189700000e8322d00/frameset.htm

[14] X14.Ahmed AlWadi, *How to create Table in SAP*, Amazon Digital Services, 2013

[15] X15.http://www.erpgreat.com/abap/the-different-types-of-sap-tables.htm

[16] X16.http://www.recercat.cat/bitstream/handle/2072/5419/PFCLopezRuizAnnex3.pdf?sequence=4

[17] X17.https://wiki.scn.sap.com/wiki/display/ABAP/ALV+easy+tutorial [x]

[18] X18.http://www.se80.co.uk/sapfms/r/reus/reuse_alv_grid_display.htm [xx]

[19] X19. Muntean, M., Advanced Business Reporting, *Proceedings of the IE 2016 International Conference*, 2016, Inforec Publishing House, pp. 126-132

[20] X20. ***, *SAP BW Training Tutorial*, http://searchsap.techtarget.com/tutorial/SAP-BW training-tutorial