Extractor for Multi Value Class Characteristic Values using Function Module

Applies to:
SAP BI 7.0, BW 3.5. For more information, visit the EDW homepage.

Summary
This article explains how to develop generic extractor which will be useful to extract multi value class characteristic data. This will be applicable to classification data of material, customer, vendor etc.

Author: Vinod Bokade
Company: Larsen & Toubro infotech Ltd.
Created on: 17 March 2011

Author Bio
Vinod Bokade is a SAP BI/BO ABAP Consultant currently working in Larsen & Toubro Infotech Ltd. He has involved in implementation, upgrade, support and enhancement of various BI/BW projects.
# Table of Contents

Introduction .......................................................................................................................................................... 3  
Business Scenario .................................................................................................................................................. 3  
Steps Involved: .................................................................................................................................................. 4  
  Create a Structure ............................................................................................................................................ 4  
  Create a View .................................................................................................................................................. 5  
  Create a Function Module ............................................................................................................................... 6  
  Create a generic extractor ................................................................................................................................. 11  
Related Content .................................................................................................................................................. 12  
Disclaimer and Liability Notice .......................................................................................................................... 13
Introduction

Class Characteristic data is very commonly used for material, customer, equipment, vendor etc. SAP provides CTBW datasource which takes care of only single characteristic value. This document outlines how to develop a generic extractor which will take care of multiple values of single characteristic. This document talks about creating a delta enabled generic extractor using function module to fetch class characteristic values data.

Business Scenario

Your company is planning to extract Material classification multi value characteristics data from ECC system to BI. This data will be used to get displayed on reports. Standard SAP CTBW extractors take care of single value of characteristic, which is not fulfilling the basic requirement i.e. multi value of a single characteristic. This can be useful for ECC reports in order to show objects with their class type, characteristics along with their values.

Data needs to be extracted:

Go To MM03 : Enter Material No, Select View as Classification and under classification tab find Class Type, Class, Characteristic and their values.

Here we are going to see, how a generic extractor using function module will be helpful to extract multi value characteristic data from ECC. In above example, there is a characteristic Language having two values as English & French, we will see how to extract these values using a generic extractor.
**Steps Involved:**

- Create a structure ZSTRUCTURE, which actually store the Class Characteristic details
- Create a view ZV_KLAH_KSSK based on KLAH KSSK table to take care of delta
- Create a function module ZFM_CLASS_CHAR_VALUE based on structure, view
- Create a generic extractor ZCLASS_CHAR_MAT based on structure and function module

**Create a Structure**

Go to Transaction code SE11, select radio button data type, and give name as per you choice, e.g. ZSTRUCTURE. This is the structure of our datasource which is going to extract class, characteristic value data. Use following fields to define a structure

<table>
<thead>
<tr>
<th>Component</th>
<th>R...</th>
<th>Component type</th>
<th>Data Type</th>
<th>Len...</th>
<th>Dec...</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJEK</td>
<td></td>
<td>OBJSN</td>
<td>CHAR</td>
<td>80</td>
<td></td>
<td>Key of object to be classified</td>
</tr>
<tr>
<td>CLASS</td>
<td></td>
<td>KLASSE_D</td>
<td>CHAR</td>
<td>18</td>
<td></td>
<td>Class number</td>
</tr>
<tr>
<td>KART</td>
<td></td>
<td>KLASSEKART</td>
<td>CHAR</td>
<td>3</td>
<td></td>
<td>Class Type</td>
</tr>
<tr>
<td>ADATU</td>
<td></td>
<td>ERDAT</td>
<td>DATE</td>
<td>8</td>
<td></td>
<td>Date on which Record Was Created</td>
</tr>
<tr>
<td>BDATU</td>
<td></td>
<td>ABDAT</td>
<td>DATE</td>
<td>8</td>
<td></td>
<td>Changed On</td>
</tr>
<tr>
<td>ARIO</td>
<td></td>
<td>DEC</td>
<td>DEC</td>
<td>2</td>
<td></td>
<td>Counter</td>
</tr>
<tr>
<td>CHARACT</td>
<td></td>
<td>ANAM</td>
<td>CHAR</td>
<td>30</td>
<td></td>
<td>Characteristic Name</td>
</tr>
<tr>
<td>VALUE_FROM</td>
<td></td>
<td>AIFLY</td>
<td>FLTP</td>
<td>16</td>
<td></td>
<td>Internal floating point from</td>
</tr>
<tr>
<td>VALUE_TO</td>
<td></td>
<td>AIFLE</td>
<td>FLTP</td>
<td>16</td>
<td></td>
<td>Internal floating point to</td>
</tr>
<tr>
<td>VALUE_RELATION</td>
<td></td>
<td>AICODE</td>
<td>CHAR</td>
<td>1</td>
<td></td>
<td>Code for value dependency</td>
</tr>
<tr>
<td>INHERITED</td>
<td></td>
<td>FLHIN</td>
<td>CHAR</td>
<td>1</td>
<td></td>
<td>Indicators: characteristic is inherited</td>
</tr>
</tbody>
</table>

**Dictionary: Display Structure**

[Diagram of Dictionary: Display Structure]

**Structure**

- Structure: ZSTRUCTURE
- Short Description: BW Structure for Class Char data

**Attributes: Components**

[Table showing structure details]

**Currency/quantity fields**

[Table showing currency/quantity fields]
Create a View

Go to transaction code SE11, select radio button view, and create a database view on KLAH and KSSK e.g. ZV_KLAH_KSSK. Use following fields and join conditions to define view, fields ADATU and VDATU will take care of delta, and logic is defined in function module (see below).
Create a Function Module

Important to know before creating a function module, Make a copy of function group (RSAX) and a function module inside it (RSAX_BIW_GET_DATA_SIMPLE) and try to make modification as follow in your function module e.g. ZFM_CLASS_CHAR_VALUE.

Give structure name (ZSTRUCTURE), which we created

and in source code use the following code

FUNCTION ZFM_CLASS_CHAR_VALUE.

**"**
**"**Local Interface:
**"**
**"**IMPORTING
**"**
**"** VALUE(I_REQUNR) TYPE SRSC_S_IF_SIMPLE-REQUNR
**"** VALUE(I_DSOURCE) TYPE SRSC_S_IF_SIMPLE-DSOURCE OPTIONAL
**"** VALUE(I_MAXSIZE) TYPE SRSC_S_IF_SIMPLE-MAXSIZE OPTIONAL
**"** VALUE(I_INITFLAG) TYPE SRSC_S_IF_SIMPLE-INITFLAG OPTIONAL
**"** VALUE(I_READ_ONLY) TYPE SRSC_S_IF_SIMPLE-READONLY OPTIONAL
**"** VALUE(I_REMOTE_CALL) TYPE SBIWA_FLAG DEFAULT SBIWA_C_FLAG_OFF
**"** TABLES
**"**
**"** I_T_SELECT TYPE SRSC_S_IF_SIMPLE-T_SELECT OPTIONAL

*"**
*"**
" I_T_FIELDS TYPE SRSC_S_IF_SIMPLE-T_FIELDS OPTIONAL
" E_T_DATA STRUCTURE ZSTRUCTURE OPTIONAL
" EXCEPTIONS
" NO_MORE_DATA
" ERROR_PASSED_TO_MESS_HANDLER

* Maximum number of lines for DB table
  STATICS: s_s_if TYPE srsc_s_if_simple,

  counter
    s_counter_datapakid LIKE sy-tabix,

  cursor
    s_cursor TYPE cursor.

* Auxiliary Selection criteria structure
  DATA: l_s_select TYPE srsc_s_select.

* Maximum number of lines for DB table
* STATICS: s_s_if TYPE srsc_s_if_simple,
* counter
*    s_counter_datapakid LIKE sy-tabix,
* cursor
*    s_cursor TYPE cursor.
* Select ranges
  RANGES: l_r_klart FOR zd_klah_kssk-klart,
          l_r_adatu FOR zd_klah_kssk-adatu,
          l_r_objek FOR zd_klah_kssk-objek.

* Data Declaration
  DATA : c_t_data TYPE zd_klah_kssk OCCURS 0 WITH HEADER LINE.
  DATA : it_allocvaluesnum TYPE bapi1003_alloc_values_num OCCURS 0 WITH HEADER LINE.
  DATA : it_allocvalueschar TYPE bapi1003_alloc_values_char OCCURS 0 WITH HEADER LINE.
  DATA : it_allocvaluescurr TYPE bapi1003_alloc_values_curr OCCURS 0 WITH HEADER LINE.
  DATA : it_return TYPE bapiret2 OCCURS 0.

  data : wa_allocvaluesnum TYPE bapi1003_alloc_values_num.
  data : wa_allocvalueschar TYPE bapi1003_alloc_values_char.
  data : wa_allocvaluescurr TYPE bapi1003_alloc_values_curr.

  data : count type ZSCLASS_CHAR-srno.

* Initialization mode (first call by SAPI) or data transfer mode
* (following calls) ?
  IF i_initflag = sbiwa_c_flag_on.

************************************************************************
* Initialization: check input parameters
*                 buffer input parameters
*                 prepare data selection
************************************************************************
* Check DataSource validity

CASE i_dsourse.

  WHEN 'ZDS_CLASS_CHAR_MAT'.
  WHEN OTHERS.
    IF 1 = 2. MESSAGE e009(r3). ENDIF.

  * this is a typical log call. Please write every error message like this
    log_write 'E'
    'message type 'R3'
    'message class '009'
    i_dsourse 'message variable 1'
    '. 'message variable 2
    RAISE error_passed_to_mess_handler.

ENDCASE.

APPEND LINES OF i_t_select TO s_s_if-t_select.

* Fill parameter buffer for data extraction calls

  s_s_if-requnr = i_requnr.
  s_s_if-dsource = i_dsourse.
  s_s_if-maxsize = '100'. 'i_maxsize.

* Fill field list table for an optimized select statement
  * (in case that there is no 1:1 relation between InfoSource fields
  * and database table fields this may be far from being trivial)

  APPEND LINES OF I_T_FIELDS TO S_S-IF-T_FIELDS.

ELSE.
  "Initialization mode or data extraction ?"

*****************************************************************************
* Data transfer: First Call OPEN CURSOR + FETCH
* Following Calls FETCH only
*****************************************************************************

* First data package -> OPEN CURSOR
  IF s_counter_datapakid = 0.

* Fill range tables BW will only pass down simple selection criteria
  * of the type SIGN = 'I' and OPTION = 'EQ' or OPTION = 'BT'.

    LOOP AT s_s_if-t_select INTO l_s_select WHERE fieldnm = 'ADATU'.
      MOVE-CORRESPONDING l_s_select TO l_r_adatu.
      APPEND l_r_adatu.
    ENDLooP.

    LOOP AT s_s_if-t_select INTO l_s_select WHERE fieldnm = 'OBJEK'.
      MOVE-CORRESPONDING l_s_select TO l_r_objek.
      APPEND l_r_objek.
    ENDLooP.

* Determine number of database records to be read per FETCH statement
  * from input parameter I_MAXSIZE. If there is a one to one relation
  * between DataSource table lines and database entries, this is trivial.
  * In other cases, it may be impossible and some estimated value has to
  * be determined.
  
  OPEN CURSOR WITH HOLD s_cursor FOR
SELECT * FROM zd_klah_kssk
WHERE klart EQ 'Z01' OR klart EQ '026' OR klart EQ '200'
  OR klart EQ '300' OR klart EQ '022' OR klart EQ '023')
  AND ( adatu >= l_r_adatu OR
       vdatu >= l_r_vdatu )
  AND objek IN l_r_objek.
ENDIF.  "First data package ?

* Fetch records into interface table.
* named E_T_'Name of extract structure'.
FETCH NEXT CURSOR s_cursor
  APPENDING CORRESPONDING FIELDS
  OF TABLE c_t_data
  PACKAGE SIZE s_s_if-maxsize.

IF sy-subrc <> 0.
  CLOSE CURSOR s_cursor.
  RAISE no_more_data.
ELSE.

  LOOP AT c_t_data.
  refresh : it_allocvaluesnum, it_allocvalueschar, it_allocvaluescurr, it_return.
  clear : it_allocvaluesnum, it_allocvalueschar, it_allocvaluescurr, it_return.

  CALL FUNCTION 'BAPI_OBJCL_GETDETAIL'
    EXPORTING
      objectkey         = c_t_data-objek
      objecttable      = 'MARA'
      classnum         = c_t_data-class
      classtype        = c_t_data-klart
    TABLES
      allocvaluesnum   = it_allocvaluesnum
      allocvalueschar  = it_allocvalueschar
      allocvaluescurr  = it_allocvaluescurr
      return           = it_return.
  IF sy-subrc = 0.
    e_t_data-objek   = c_t_data-objek.
    e_t_data-class   = c_t_data-class.
    e_t_data-klart   = c_t_data-klart.
    e_t_data-adatu   = c_t_data-adatu.
    e_t_data-vdatu   = c_t_data-vdatu.

    IF it_allocvaluesnum IS NOT INITIAL.
      LOOP AT it_allocvaluesnum.

      MOVE it_allocvaluesnum TO wa_allocvaluesnum.

      AT NEW CHARACT.
        CLEAR count.
        MOVE '1' TO count.
      ENDAT.

      MOVE-CORRESPONDING wa_ALLOCVALUESNUM TO E_T_DATA.
      MOVE count TO e_t_data-srno.
APPEND e_t_data.
count = count + 1.
ENDLOOP.
ENDIF.

IF it_allocvalueschar IS NOT INITIAL.
LOOP AT it_allocvalueschar.
MOVE it_allocvalueschar TO wa_allocvalueschar.

AT NEW CHARACT.
  CLEAR count.
  MOVE '1' TO count.
ENDAT.

MOVE-CORRESPONDING it_allocvalueschar TO e_t_data.
MOVE count TO e_t_data-srno.
APPEND e_t_data.
count = count + 1.
ENDLOOP.
ENDIF.

IF it_allocvaluescurr IS NOT INITIAL.
LOOP AT it_allocvaluescurr.
MOVE it_allocvaluescurr TO wa_allocvaluescurr.

AT NEW CHARACT.
  CLEAR count.
  MOVE '1' TO count.
ENDAT.

MOVE-CORRESPONDING it_allocvaluescurr TO e_t_data.
MOVE count TO e_t_data-srno.
APPEND e_t_data.
count = count + 1.
ENDLOOP.
ENDIF.
ENDIF.

ENDLOOP.
ENDIF.

s_counter_datapakid = s_counter_datapakid + 1.
ENDIF. "Initialization mode or data extraction ?

ENDFUNCTION.
Create a generic extractor

Now finally create the generic datasource as ZDS_CLASS_CHAR_MAT using T. Code RSO2 radio button Transaction data.

Select Applic. Component as MM, give some descriptions
Function Module :    ZFM_CLASS_CHAR_VALUE
Extract.Struct. :   ZSTRUCTURE

Check datasource result using transaction code RSA3, we will get the following result

Note: We can use this datasource for other classification data also, by making small modification in source code like changing table name. In above example, we used table MARA for material classification (search for ‘MARA’ in source code and change it to ‘KNA1’ so that it can be used for Customer Classification data. Take help of ABAPer for any difficulties during development of function module.
Related Content

http://wiki.sdn.sap.com/wiki/display/Snippets/Fetch+material+Classifications+and+characteristics

For more information, visit the EDW homepage
Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.