How to Create a Generic Data Source Based on a Custom View and Further Enhance the Data Source for BW Extraction

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

Summary
The document will help us understand how to generate a generic data source based on a view created by joining two tables. The data source will be further enhanced to enable extraction of additional fields from other relevant tables.

Author: Umesh Pednekar
Company: Howdens Joinery Co.
Created on: 17 November 2010

Author Bio
Umesh is working as a SAP BI/BW Analyst in Howdens Joinery co and has an techno-functional experience of over 6 years in designing, developing and implementing/upgrading SAP BW-BI solutions across various Industries in India and in the UK.
How to Create a Generic Data Source Based on a Custom View and Further Enhance the Data Source for BW Extraction

Table of Contents
Overview..........................................................................................................................................................3
Creating a View Based on the DB Tables ........................................................................................................3
Creating a Generic Data source based on the View .........................................................................................8
Checking the data in the Generic Data source using the Extractor Checker ...............................................11
Enhance the generic data source extractor with additional fields – in the appended structure ......................13
Un-hiding the fields in the appended structure of a data source ..................................................................18
Populating Appended fields in the extract structure of a Data source (through Customer Exit) ..................20
Checking the data in the Custom fields of the Generic Data source using the Extractor Checker ..............23
Appendix..........................................................................................................................................................24
Related Content..............................................................................................................................................27
Disclaimer and Liability Notice.....................................................................................................................28
Overview

Purpose of this document is to help us understand how to create a custom view based on transparent tables and create a generic data source based on the view. The document also describes the steps involved in enhancing the generic data source created to populate it with fields from other relevant tables.

The document considers a very unique situation which calls for extraction of Purchasing Info records from tables EINA and EINE.

The document explain step by step processes to execute the following activities:

- Creating a view based on the tables.
- Create a generic data source based on the view.
- Check the data in the newly created generic data source.
- Enhance the generic data source extractor with additional fields – in the appended structure.
- Unhide the fields in the appended structure.
- Populate the fields in the appended structure through an ABAP routine.
- Check the data in the data source – along with the appended fields which are now populated.

The entire set of activities are performed on the source system.

Creating a View Based on the DB Tables

Check the tables that you wish to create the join on. We are considering tables EINA and EINE. Please remember that EINE is not a table but a template and thus we are creating a join. Our objective is to obtain details of Purchasing info records and then establish a link between the info records and other tables viz. Condition tables (KONP, KONH, EORD), purchase transaction tables (EKKO, EKPO).

EINA is Purchase Info record: General data table, EINE is Purchasing Info record: Purchasing Organization data table.

We create a join on these tables and build a generic data source on the join. The generic data source will then be enhanced with fields that we need additionally (and which will be populated from other tables mentioned above).

In SAP R/3 or ECC enter Transaction Se11 which enables you to create a table or view. We are using the name ZMMPUR_INFOREC.
Click ‘Create’

Select the view type. We are creating a database view.

The system now displays the Maintenance view screen.

In the Table/Join Conditions tab, enter the tables that you wish to build the join on.

Enter a description for the View.

We now need to establish a relationship between these two tables i.e. identify fields that would integrate them with each other.
If you are unsure of this activity, then select the tables involved in the join and select ‘Relationships’ and the system will automatically prompt you the relationship link between these tables.

The relationship is established between the two tables via fields MANDT and INFR.

In the ‘View Fields’ tab enter the fields that you wish to include in the View. These fields will be from either of the tables on which the join is created.
Please remember to include the 'key fields' from each of the tables in the fields selection else you will encounter an error during activation of the view.

Key fields for EINA is INFNR – Info Record and for EINE are INFNR (Info record), EKORG (Purchasing Organization), ESOKZ (Info Category) and WERKS (Plant).

In the ‘Selection Conditions’ tab, enter the specific condition/s that you wish to apply during view creation. For eg. You may want to consider only active records and ignore deleted records. In that case, the selection condition would be:

- **<Table name>** = to which the field belongs
- **<Field name>=** field on which you want to apply the condition
- **<Operator>=** Applicable operator from the list
- **<Comparative Value>=** value of the Field that needs to be applied during view creation.

You can include multiple selections by using the <AND/OR> operators.

Save the View.
You will be prompted to enter the development class as well as Transport request # for the development that you are carrying. Enter the relevant values.
Click on the ‘Check’ button to check the View for inconsistencies.

Activate the View by clicking on the Activate icon.

The view ZMMPUR_INFOREC is now activated.

Creating a Generic Data source based on the View

We will now proceed to the step of creating a Generic data source based on the view we just created.

Use Transaction Code RSO2 to create a Generic data source.

Enter the name of the data source you wish to create. We are using ZMMPUR_INFOREC. Remember, this is a transaction data source. ‘Create’ the data source.
On the Create <data source> screen, enter the parameters as required.

**Application Component:** Component name where you wish to place the data source in the App. Component hierarchy.

**Text:** Descriptions (Short, Medium and Long) for the data source.

**View/Table:** Name of the Table/View on which you wish to create the Generic data source. In our case it is ZMMPUR_INFOREC.

You can also create a generic data source by extraction data through Infoset Queries and Function modules.

Save the data source.

Enter the Development class followed by the request number.

The Generic datasource is now displayed allowing you to Select as well as Hide field. The fields ‘hidden’ will not be available for extraction. Fields in the ‘Selection’ tab will be available for Selection in the Infopackage during data extraction from the source system to the PSA.
How to Create a Generic Data Source Based on a Custom View and Further Enhance the Data Source for BW Extraction

Select the relevant fields and Save the data source.

Data source is saved successfully.
Checking the data in the Generic Data source using the Extractor Checker

You can check the data in the data source by using the Extractor checked – Tcode RSA3.

Enter the data source name which we created in the earlier step ie. ZMMPUR_INFOREC and click on ‘Extraction’.

The available no. of records as per the initial settings on RSA3 is available for viewing.

Select ‘Continue’ and on the next screen select ‘Display list’ to display the number of data packets available.

Select any one of the data packets and double-click.
The records in the data packet are displayed.
Enhance the generic data source extractor with additional fields – in the appended structure

The next step is to enhance the generic data source with fields that we need to populate from other tables. In this case, we need to pull data from Conditions tables viz. KONP, KONH, KONV, EORD as well as Text tables for conditions.

Fields that we need are:

<table>
<thead>
<tr>
<th>#</th>
<th>Field Name</th>
<th>Description</th>
<th>Table</th>
<th>Source Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>KSCHL</td>
<td>Condition type</td>
<td>KONH / KONP / KONV</td>
<td>KSCHL</td>
</tr>
<tr>
<td>2</td>
<td>FESKZ</td>
<td>Indicator: Fixed vendor</td>
<td>EORD</td>
<td>FLIFN</td>
</tr>
<tr>
<td>3</td>
<td>KDATB</td>
<td>Valid-from date</td>
<td>KONH / EKKO</td>
<td>DATAB</td>
</tr>
<tr>
<td>4</td>
<td>KDATE</td>
<td>Valid to date</td>
<td>KONH / EKKO</td>
<td>DATBI</td>
</tr>
<tr>
<td>5</td>
<td>VTEXT</td>
<td>Name</td>
<td>T685T</td>
<td>VTEXT</td>
</tr>
<tr>
<td>6</td>
<td>KPEIN</td>
<td>Condition pricing unit</td>
<td>KONP / KONV</td>
<td>KPEIN</td>
</tr>
<tr>
<td>7</td>
<td>KMEIN</td>
<td>Condition unit</td>
<td>KONP / KONV</td>
<td>KMEIN</td>
</tr>
<tr>
<td>8</td>
<td>KBETR</td>
<td>Rate (condition amount or percentage) where no scale exists</td>
<td>KONP / KONV</td>
<td>KBETR</td>
</tr>
<tr>
<td>9</td>
<td>KONWA</td>
<td>Rate unit (currency or percentage)</td>
<td>KONP / KONV</td>
<td>KONWA</td>
</tr>
</tbody>
</table>

Use Transaction code **RSA2** to enhance any data source.

Enter the data source name we created in the earlier session and click ‘Display’.
Select the 'Extract Structure' i.e. ZOXED10108 and double-click to view the structure of the data source.

We will edit the structure and Append relevant fields for population.

Click on the <Display-Change> icon

You are now ready to append the data source structure.
In the ‘GoTo’ menu, select ‘Append structure’ as shown below.
The system prompts you to enter the name of the Append Structure.
Select ‘continue’ to get the Append Structure screen.

Enter the relevant fields in the ‘Component’ column and the name of the Structure in Short text. Please remember that these fields are Custom-fields and should be within defined rules of the Customer namespace.

Enter the Component Type for each of the fields – PI refers the base tables for these values.
In the Currency / Quantity fields tab, Add reference field and Table for the Quantity/Currency fields.

What have we done?

We added the Currency field i.e. KBETR as well as the Currency key field KONWA as component types to ZZKBETR and ZZKONWA. As a reference to the currency field i.e. ZZKBETR I have added the ZZKONWA field from the current structure i.e. ZOXED10108. (Pl remembers we are not using the Original table name here as it does not contain the field KONWA or ZZKONWA).

Save the structure and add it in the relevant development class and transport request.

Activate the datasource.

Check the structure for inconsistencies.

Go back to the initial screen and click on the Extract structure name i.e. ZOXED10108 in this case.

The extract structure of the data source is displayed. The view now includes the appended structure.
**How to Create a Generic Data Source Based on a Custom View and Further Enhance the Data Source for BW Extraction**

You can access the data source through the extractor checker – Tcode RSA3 – and check for the data. You will find that the appended fields are not displayed. Why? The appended fields are still ‘hidden’ and unless we ‘unhide’ them the fields will not appear in the extractor checker.

### Appended structure

<table>
<thead>
<tr>
<th>Component</th>
<th>Component type</th>
<th>Length</th>
<th>Data type</th>
<th>Short text</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPEND</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXZEEB1001</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXZEEB1002</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXZEEB1003</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXZEEB1004</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXZEEB1005</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can access the data source through the extractor checker – Tcode RSA3 – and check for the data. You will find that the appended fields are not displayed. Why? The appended fields are still ‘hidden’ and unless we ‘unhide’ them the fields will not appear in the extractor checker.
**Un-hiding the fields in the appended structure of a data source**

To unhide fields from the data source structure go to transaction **RSA6**.

Select the data source we created from the application component hierarchy. (We selected MM as the application component for the data source).

Click on ‘Change data source’ icon from the top menu

The data source fields are displayed. The newly appended fields are displayed with a tick-mark in the 'Hide field' column. Fields if hidden in the extraction structure are excluded from the data transfer, set this flag. The field is then no longer available in BW for determining transfer rules and therefore cannot be used for generating transfer structures.

Unhide the fields.

The indicator **Field known only in Exit** is set for the fields in an append structure, meaning that by default, these fields are not passed to the extractor in the field list and the selection table. Remove the **Field known only in Exit** indicator, to get the Service API to pass the field in the append structure to the extractor.

Save the data source.
If you now check the Extractor checker (RSA3) you will find the appended fields, however, without any values.

The appended fields which do not pull any value need to be populated through a Customer Exit. The customer exit will integrate the data source / extract structure with other relevant tables and populate these fields.
Populating Appended fields in the extract structure of a Data source (through Customer Exit)

The next step is to populate the custom fields that we have included in the Generic data source. These custom fields could be Z-fields or standard fields from SAP. You will need to write a ABAP customer exit to populate these fields.

Use T-code CMOD to access the Management screen for custom enhancements.

![Project Management of SAP Enhancements](image)

Enter the relevant Project name and click on change.

![Project Management of SAP Enhancements](image)

At the next prompt, select ‘Continue’

![Project Management of SAP Enhancements](image)
The standard SAP enhancements provided by SAP for BW extraction enhancements are displayed.

Select the enhancement RSAP00001 and component.

The system now prompts you to select the relevant customer exit.

As we are enhancing a transaction data source we will be selecting the relevant customer exit that is meant for populating the transaction data sources. This customer exit is EXIT_SAPLRSAP_001. Take the cursor on the exit and double-click.

Select the Include (ZXRSAU01).
The ABAP Editor is displayed. You can now develop a custom routine to integrate the generic data source/extractor with other tables and fill the custom fields in the generic data source with values from other tables (please refer the table list above).

A sample code is attached in Appendix A which populates a few fields that we have added in our generic data source.

Save the exit and activate it.
Checking the data in the Custom fields of the Generic Data source using the Extractor Checker

You can check the data in the custom fields of the data source by using the Extractor checked – Tcode RSA3.
Appendix

This is an example of a code sample: The code populates a few of the custom fields that we have added in the generic extractor.

```plaintext
//This is a code sample block

CASE i_datasource.
  WHEN 'ZMM_BWDS_CONTRACTS'.
    it_contr[] = c_t_data[].

    PERFORM extract_conditions CHANGING it_contr[].

    REFRESH c_t_data.
    c_t_data[] = it_contr[].
  ENDCase.

*& ---------------------
* & Form  extract_conditions
* & --------------------------------------------------
* & Condition header KONH
* & condition details KONP
* & Conditions Transaction KONV
* & EORD Purchasing Source List
* & --------------------------------------------------------------------
* & --> p_t_data - Data from view without condition data
* &  <-- p_t_data - Data with condition data sent back to program
* & --------------------------------------------------

FORM extract_conditions CHANGING p_t_contr TYPE it_contr.

* Constants
CONSTANTS:
  lc_% TYPE c VALUE '%',
  lc_a TYPE c VALUE 'A',
  lc_m TYPE c VALUE 'M',
  lc_x TYPE c VALUE 'X'.

* Local types
TYPES:
  * For contract header list
    BEGIN OF ty_konh_vak,
    vakey TYPE vakey,
    END OF ty_konh_vak,
  * For EORD records
    BEGIN OF ty_eord ,
    matnr TYPE eord-matnr ,
    werks TYPE eord-werks ,
    ebeln TYPE eord-ebeln ,
    ebelp TYPE eord-ebelp ,
    flifn TYPE eord-flifn ,"(Fixed Vendor)
    fresw TYPE eord-fresw ,"(Fixed Issuing plant)
    febel TYPE eord-febel ,"(Fixed Purchasing Agreement)
    END OF ty_eord ,
  * For condition header details
    BEGIN OF ty_konh,
    knumh TYPE knumh,
    vakey TYPE vakey,
    datab TYPE datab,
    datbi TYPE datbi,
    END OF ty_konh,
  * For condition item details
    BEGIN OF ty_konp,
    knumh TYPE knumh,
    kschl TYPE kschl,
    kbetr TYPE kbetr,
    konwa TYPE konwa,
```
kpein TYPE kpein,
    kmein TYPE kmein,
END OF ty_konp.
* local data
DATA:
    lv_flag    TYPE c,
    lw_contr   TYPE zoxed10108,
    lt_contr   TYPE TABLE OF zoxed10108 WITH HEADER LINE,
    lt_konh    TYPE TABLE OF ty_konh WITH HEADER LINE,
    lt_konh_vak TYPE TABLE OF ty_konh_vak WITH HEADER LINE,
    lw_konp    TYPE ty_konp,
    lt_konp    TYPE TABLE OF ty_konp WITH HEADER LINE,
    lt_eord    TYPE TABLE OF ty_eord WITH HEADER LINE,
    lt_final   TYPE TABLE OF zoxed10108 WITH HEADER LINE.
* loop at the view data and build the range (like cursor)
LOOP AT p_t_contr INTO lw_contr.
* Build the range values
  CONCATENATE lw_contr-ebeln lw_contr-ebelp INTO lt_konh_vak-vakey.
* add to the varkey table
  APPEND lt_konh_vak.
ENDLOOP.
* Select EORD records
SELECT matnr werks ebeln ebelp flifn febel fresw
FROM eord
INTO CORRESPONDING FIELDS OF TABLE lt_eord
FOR ALL ENTRIES IN p_t_contr
WHERE matnr EQ p_t_contr-matnr
  AND ebeln EQ p_t_contr-ebeln.
break yesudasf.
* Now for every entry in the P_T_CONTR table check if there is entry in
  EORD table and if it has more than one entry present for different
  plants then repeat the record for the plant details.
LOOP AT p_t_contr INTO lw_contr.
  CLEAR lt_contr.
  MOVE CORRESPONDING lw_contr TO lt_contr.
  CLEAR lv_flag.
  LOOP AT lt_eord WHERE matnr EQ lt_contr-matnr
    AND ebeln EQ lt_contr-ebeln
    AND ebelp EQ lt_contr-ebelp.
    lt_contr-werks = lt_eord-werks.
  * determine if Fixed vendor of supply
    IF lt_eord-flifn EQ lc_x "(Fixed Vendor)
      OR lt_eord-fresw EQ lc_x "(Fixed Issuing plant)
      OR lt_eord-febel EQ lc_x ".(Fixed Purchasing Agreement)
    lt_contr-feskz = 1c_x. " mark as Fixed vendor is present
    ENDIF.
    APPEND lt_contr.
    CLEAR lt_contr-feskz.
    lv_flag = 1c_x.
  ENDLOOP.
  IF lv_flag IS INITIAL.
    APPEND lt_contr.
  ENDIF.
ENDLOOP.
* Now Get condition header records for the LT_KONH_VAK entry
SELECT knumh vakey datab datbi
FROM konh
INTO TABLE lt_konh
FOR ALL ENTRIES IN lt_konh_vak
WHERE vakey = lt_konh_vak-vakey.
IF sy-subrc EQ 0.
  * As there could be many conditions with different validity dates hence
* get the most latest record by sorting it based on the date.
  
  ```
  SORT lt_konh DESCENDING BY knumh datab ASCENDING vakey.
  ```

* For each entry in the condition header get the condition details from
  the `KONP` table where the condition is active

```
SELECT knumh kschl kbetr konwa kpein kmein
  FROM konp
  INTO TABLE lt_konp
  FOR ALL ENTRIES IN lt_konh
  WHERE knumh EQ lt_konh-knumh
  AND loevm_ko NE lc_x.
ENDIF.
```

```
LOOP AT lt_contr INTO lw_contr.
```

* Go find the condition header record for the contract from the
  `KONH` table

```
CLEAR lv_flag.
READ TABLE lt_konh WITH KEY
  vakey+0(10) = lw_contr-ebeln
  vakey+10(5) = lw_contr-ebelp.
```

* Get the condition GROSS amount first as this is used to calculate the
  amount for percentage based condition entries

```
CLEAR lw_konp.
READ TABLE lt_konp INTO lw_konp
  WITH KEY
    knumh = lt_konh-knumh
    kschl = 'YB00'.
```

* Now get all other condition details

```
LOOP AT lt_konp WHERE knumh EQ lt_konh-knumh.
```

* Get the condition text

```
CLEAR lw_contr-vtext.
SELECT SINGLE vtext
  FROM t685t
  INTO lw_contr-vtext
  WHERE
    spras EQ sy-langu
    AND kvewe EQ lc_a
    AND kappl EQ lc_m
    AND kschl EQ lt_konp-kschl.
```

* Assign all other values

```
lw_contr-kschl = lt_konp-kschl.
lw_contr-konwa = lt_konp-konwa.
```

* Calculate the actual amount value for any percentage figure

```
IF lt_konp-konwa EQ lc_%.
  lw_contr-kbetr = (lw_konp-kbetr * lt_konp-kbetr) / 1000.
ELSE.
  lw_contr-kbetr = lt_konp-kbetr.
ENDIF.
```

```
lw_contr-kpein = lt_konp-kpein.
lw_contr-kmein = lt_konp-kmein.
lt_final = lw_contr.
APPEND lt_final.
```

```
lv_flag = lc_x.
CLEAR :lw_contr-kschl, lw_contr-konwa, lw_contr-kbetr,
       lw_contr-kpein, lw_contr-kmein.
```

```
ENDLOOP.
```

```
ENDIF.
```

```
IF lv_flag IS INITIAL.
  lt_final = lw_contr.
  APPEND lt_final.
ENDIF.
ENDLOOP.
```

```
SORT lt_contr.
```

```
DELETE ADJACENT DUPLICATES FROM lt_contr.
```

```
REFRESH p_t_contr.
```

```
 p_t_contr[] = lt_final[].
```

ENDFORM.
Related Content

SAP Network Blog: Extract Checker
SAP Network Blog: Generic extraction using a function module
SAP Wiki: Frequently Used Tables
SAP Network Blog: Generic Data extraction in SAP BW decoded – Part 1
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.