How to Improve the Performance of the Extractor - Cross Company Stock in Transit

Applies to:

BW >= 2.0B / R/3 >= 4.0B. For more information, visit the EDW homepage

Summary

There is already a 'How to do' document on reporting 'Cross Company Stock in Transit'. However, the extractor code might give performance issues if implemented exactly as the same way as mentioned in the original 'How to' document.

The purpose of this article is to propose optimized extraction logic for this reporting requirement.

Author: Krishnan Rajamani

Company: Capgemini

Created on: 23rd July 2011

Author Bio

Krishnan Rajamani is working with Capgemini India Pvt Limited and he has vast experience in SAP BW and SAP MM.

Table of Contents

Background 3
Introduction 3
The Step By Step Solution 3
Overview of the steps 4
Steps in ECC 4
Steps in BW 18
Related Content 19
Disclaimer and Liability Notice 20

Background

Cross company stock in transit information is necessary to get complete 360 degree view of the stock position. The current BW stock reporting solution does not cover this scenario. However there is a 'How to do' document on this topic explaining step by step procedure of implementing this extractor.

Please refer the following link for accessing the 'how to do' document.

How to Report on Cross Company Stock in Transit

But the extractor created based on above 'How to do' document might have issues with extraction performance and also there is a small glitch in the key date used for currency conversion.

This article addresses above concerns and should be read in conjunction with original 'How to do' document.

Introduction

The original extractor runs based on the mandatory selection on 'posting date' and selects all the table information from EKKO, EKPO and EKBE for Stock in transit orders where posting date is less than or equal to the selection date.

Using this information 'Issued quantity' and 'Received Quantity' against Stock in transit orders are calculated and compared to finally arrive at stock in transit quantity.

In the new extraction method, a mandatory selection on the purchase order type is introduced and this improves the selection performance from the purchase order tables.

Secondly it is important to know that aggregated values of 'Receipt quantity' and Issued quantity' at Purchase order schedule line level are also stored in table EKET at any given point in time.

Since this extraction gives only snapshot information of stock in transit, it is better to select the data from EKET to get current position of stock in transit instead of table EKBE. Since EKET is much smaller in comparison to EKBE, extraction performance would get significantly improved in such cases.

However, this extractor also has a provision to enter posting date in the selection condition (Single Value, Not Mandatory). In general, this extractor has to be run without giving any selections on the posting date, so that it gets the information from EKET. If executed by inputting a posting date, then it fetches information from EKBE and this feature can be used to correct the stock information for any historical snapshot, still with much improved performance.

This date will also be used as the key date for currency conversion to report the stock in transit value at the company code currency.

The Step By Step Solution

Instead of trying to compare the coding level changes done between old and new extraction methods, this document will look to cover the development of new extractor independently. If the old extractor is already implemented with the naming conventions used in the original document in your system, it is advisable to create a new extractor based on the technical names given in this document. This would give the opportunity to compare the performance and data between both the extractors.

This extraction logic assumes that in the given business context, it is possible to identify the purchase document types linked to cross company stock in transit scenario, as this forms the mandatory selection condition of this extractor. If that is not the case, additional coding has to be done to get relevant purchase document types.

It is also highly recommended to have secondary indexes created on purchase document type(BSART) in table EKKO for improved performance. However this extractor has delivered better performance, even without such secondary index.

Overview of the steps

ECC

- Creation of Extract structure.
- Creation of function module extractor.
- Creation of DataSource

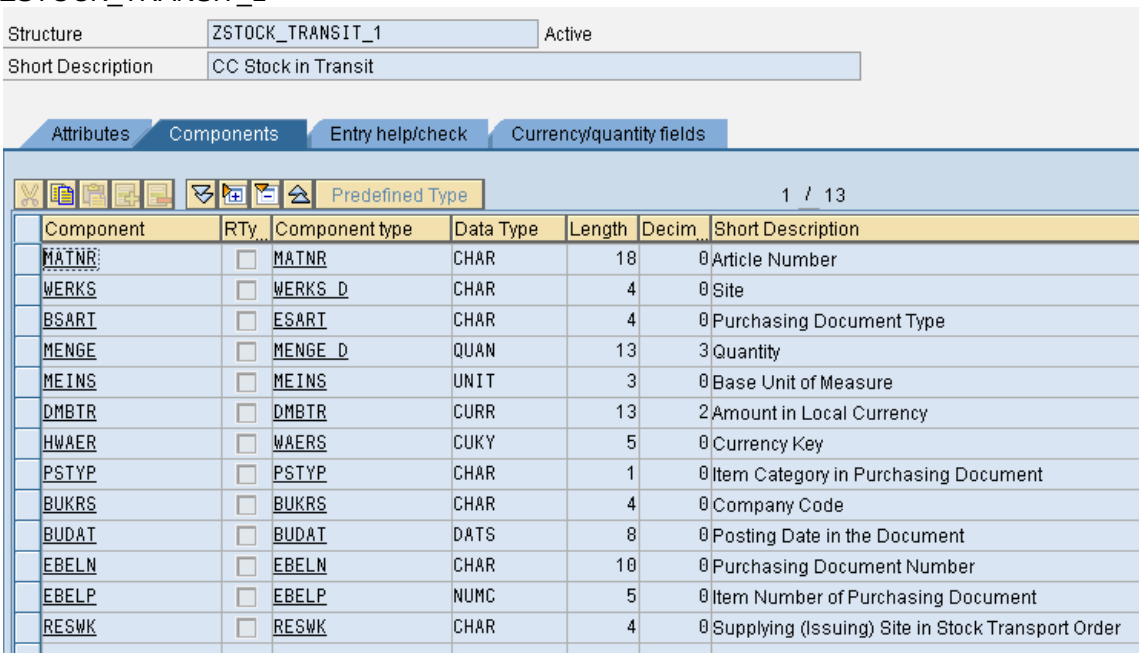
BW

- Replicating the DSO and Creation of Infopackage
- Transformation between DSO and Datasource

Steps in ECC

1) Creation of Extract structure in Tcode SE11

ZSTOCK_TRANSIT_1



Structure: ZSTOCK_TRANSIT_1 Active

Short Description: CC Stock in Transit

Attributes | Components | Entry help/check | Currency/quantity fields

Predefined Type 1 / 13

Component	RTy	Component type	Data Type	Length	Decim	Short Description
MATNR		MATNR	CHAR	18	0	Article Number
WERKS		WERKS_D	CHAR	4	0	Site
BSART		ESART	CHAR	4	0	Purchasing Document Type
MENGE		MENGE_D	QUAN	13	3	Quantity
MEINS		MEINS	UNIT	3	0	Base Unit of Measure
DMBTR		DMBTR	CURR	13	2	Amount in Local Currency
HWAER		WAERS	CUKY	5	0	Currency Key
PSTYP		PSTYP	CHAR	1	0	Item Category in Purchasing Document
BUKRS		BUKRS	CHAR	4	0	Company Code
BUDAT		BUDAT	DATS	8	0	Posting Date in the Document
EBELN		EBELN	CHAR	10	0	Purchasing Document Number
EBELP		EBELP	NUMC	5	0	Item Number of Purchasing Document
RESWK		RESWK	CHAR	4	0	Supplying (Issuing) Site in Stock Transport Order

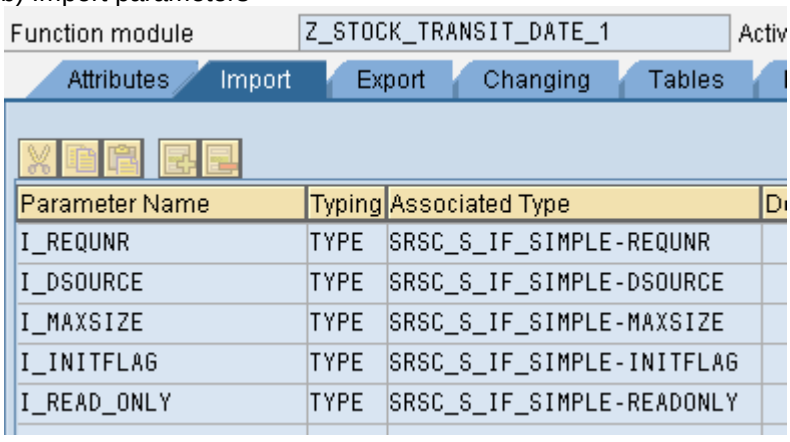
2) Creation of function module extractor

1. Function group ZZ_STOCK_TRANSIT_1
2. Function module Z_STOCK_TRANSIT_DATE

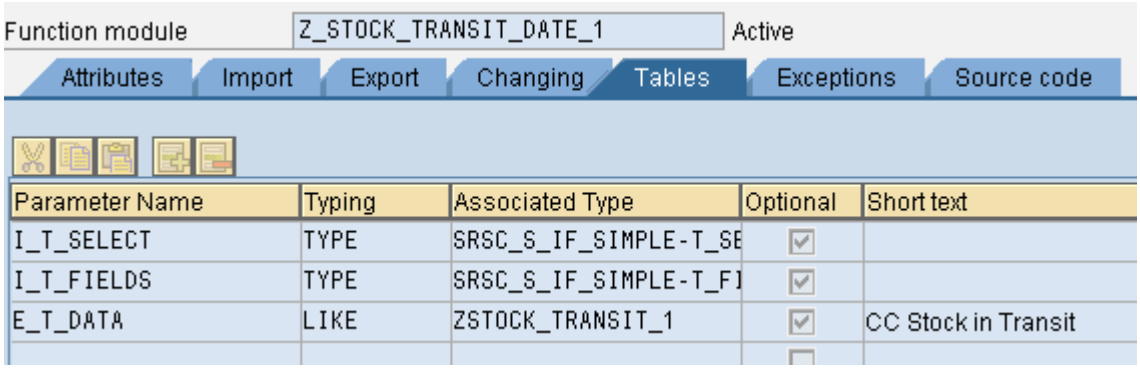
a) Global data

```
FUNCTION-POOL ZZ_STOCK_TRANSIT_1.           "MESSAGE-ID ..

TYPE-POOLS: SBIWA,
            RSAP,
            SRSC,
            RSAOT,
            RSAZT,
            ZSIT.

types: begin of po_num,
        ebeln type ebeln,
        bsart type bsart,
        INCO1 type inco1,
        INCO2 type inco2,
        wkurs type wkurs,
        bedat type bedat,
        ekgrp type ekgrp,
        ekorg type ekorg,
        lifnr type lifnr,
        lifre type lifre,
        waers type waers,
       end of po_num.


types: po_num_t type standard table of po_num.

types: begin of numki_range,
        NRNR type NRNR,
        range type ref to data,
       end of numki_range.

data: gt_select     type SRSC_T_SELECT,
      gt_fields     type SRSC_T_FIELDS,
      gt_po_fields  type SRSC_T_FIELDS,
      gt_sel_fields type SRSC_T_FIELDS.
```

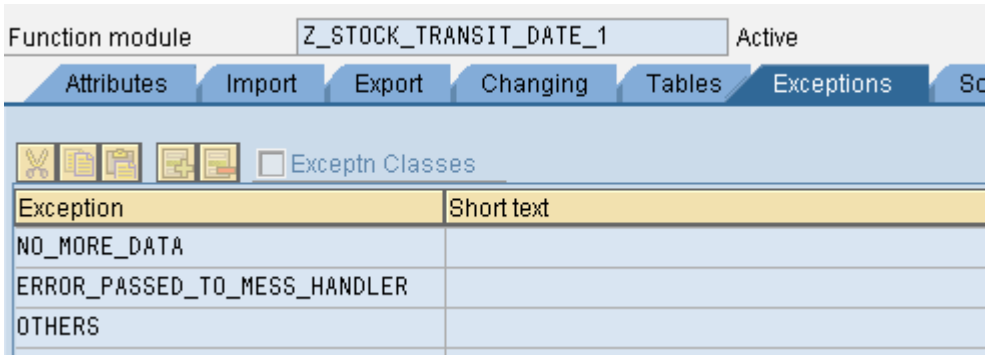
b) Import parameters



c) Tables parameters



d) Exceptions



e) Source code

```
FUNCTION Z_STOCK_TRANSIT_DATE_1.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(I_REQUNR) TYPE  SRSC_S_IF_SIMPLE-REQUNR
*"     REFERENCE(I_DSOURCE) TYPE  SRSC_S_IF_SIMPLE-DSOURCE OPTIONAL
*"     REFERENCE(I_MAXSIZE) TYPE  SRSC_S_IF_SIMPLE-MAXSIZE OPTIONAL
*"     REFERENCE(I_INITFLAG) TYPE  SRSC_S_IF_SIMPLE-INITFLAG OPTIONAL
*"     REFERENCE(I_READ_ONLY) TYPE  SRSC_S_IF_SIMPLE-READONLY OPTIONAL
*"  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  ZSTOCK_TRANSIT_1 OPTIONAL
*"  EXCEPTIONS
*"      NO_MORE_DATA
*"      ERROR_PASSED_TO_MESS_HANDLER
*"      OTHERS
*"----------------------------------------------------------------------

DATA: lr_ebeln TYPE RANGE OF ebeln,
      lr_bsart TYPE RANGE OF bSART,
      lr_werks TYPE RANGE OF werks,
      lr_budat TYPE RANGE OF budat.
```

```
IF NOT i_initflag IS INITIAL.

    gt_select = i_t_select[].
    gt_fields = i_t_fields[].
    PERFORM budat_validate CHANGING gt_select.
ELSE.
* transfer selection parameters in ranges
    PERFORM range_fill USING zsit_gc_fn_ebeln
                             gt_select
                             CHANGING lr_ebeln.
    PERFORM range_fill USING zsit_gc_fn_bsart
                              gt_select
                              CHANGING lr_bsart.
    PERFORM range_fill USING zsit_gc_fn_werks
                              gt_select
                              CHANGING lr_werks.
    PERFORM range_fill USING zsit_gc_fn_budat
                              gt_select
                              CHANGING lr_budat.
    PERFORM data_selection_transit_date USING i_maxsize
                              lr_ebeln
                              lr_bsart
                              lr_werks
                                  lr_budat
                              CHANGING e_t_data[].

ENDIF.



ENDFUNCTION.
```

f) Include LZZ_STOCK_TRANSIT_1F01

```
*---------------------------------------------------------------------*
***INCLUDE LZZ_STOCK_TRANSIT_1F01 .
*---------------------------------------------------------------------*
*&---------------------------------------------------------------------*
*&      Form  BUDAT_VALIDATE
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*      <--P_GT_SELECT  text
*---------------------------------------------------------------------*

FORM build_range USING iv_sign
                       iv_option
                       iv_low
                       iv_high
                       CHANGING cr_range TYPE STANDARD TABLE.

  DATA: lp_data TYPE REF TO data.

  FIELD-SYMBOLS: <ls_fields> TYPE ANY,
                 <ls_range> TYPE ANY,
                 <lv_sign> TYPE ANY,
                 <lv_option> TYPE ANY,
```

```
                <lv_low> TYPE ANY,
                <lv_high> TYPE ANY,
                <lv_fieldname> TYPE ANY.

  CREATE DATA lp_data LIKE LINE OF cr_range.
  ASSIGN lp_data->* TO <ls_range>.
  ASSIGN COMPONENT 'SIGN' OF STRUCTURE <ls_range> TO <lv_sign>.
  ASSIGN COMPONENT 'OPTION' OF STRUCTURE <ls_range> TO <lv_option>.
  ASSIGN COMPONENT 'LOW' OF STRUCTURE <ls_range> TO <lv_low>.
  ASSIGN COMPONENT 'HIGH' OF STRUCTURE <ls_range> TO <lv_high>.

  <lv_sign> = iv_sign.
  <lv_option> = iv_option.
  <lv_low> = iv_low.
  <lv_high> = iv_high.

  INSERT <ls_range> INTO TABLE cr_range.

ENDFORM. " build_range

***************************

FORM budat_validate CHANGING ct_select TYPE srsc_t_select.

  DATA: ls_select TYPE srsc_s_select.
  DATA: lt_select TYPE srsc_t_select.
  DATA: ls_return TYPE bapiret2.
  DATA: lv_tabix TYPE sytabix.

  lt_select = ct_select.

  SORT lt_select BY fieldnm.

  DELETE lt_select WHERE fieldnm NE zsit_gc_fn_budat.

  DESCRIBE TABLE lt_select.

  IF sy-tfill > 1.
    MESSAGE e113(zstocktransit) WITH zsit_gc_fn_budat
       RAISING error_passed_to_mess_handler.

*Selection parameter & only allows single value
  ENDIF.

  IF NOT ls_select-low IS INITIAL AND NOT ls_select-high IS INITIAL
     AND ls_select-low NE ls_select-high .
    MESSAGE e115(zstocktransit) WITH zSIT_gc_fn_budat.

*For parameter & no intervel is allowed
  ENDIF.

  READ TABLE ct_select INTO ls_select
    WITH KEY fieldnm = zsit_gc_fn_budat.
  if sy-subrc eq 0.

    lv_tabix = sy-tabix.

    IF NOT ls_select-high IS INITIAL.
      CLEAR: ls_select-high.
    ENDIF.

    ls_select-option = 'LE'.

    MODIFY ct_select FROM ls_select INDEX lv_tabix.
  endif.
```

```
ENDFORM. " budat_validate




*&---------------------------------------------------------------------*
*&      Form  data_selection_transit_date
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->IV_MAXSIZE text
*      -->IR_EBELN   text
*      -->IR_BSART   text
*      -->IR_WERKS   text
*      -->IR_BUDAT   text
*      -->CT_DATA    text
*----------------------------------------------------------------------*
FORM data_selection_transit_date USING iv_maxsize
                                       ir_ebeln TYPE STANDARD TABLE
                                       ir_bsart TYPE STANDARD TABLE
                                       ir_werks TYPE STANDARD TABLE
                                       ir_budat TYPE STANDARD TABLE
                                       CHANGING ct_data TYPE STANDARD TABLE.

  DATA: lr_plant TYPE RANGE OF werks_d,
        lr_ebeln TYPE RANGE OF ebeln,
        lr_ebelp TYPE RANGE OF ebelp,
        lr_etenr TYPE RANGE OF etenr.

  Data  :  BEGIN OF LI_EKKO,
             EBELN type ekko-ebeln,
             BUKRS type ekko-BUKRS,
             BSTYP type ekko-BSTYP,
             BSART type ekko-BSART,
             BSAKZ type ekko-BSAKZ,
             WAERS type ekko-WAERS,
             WKURS type ekko-WKURS,
             KUFIX type ekko-KUFIX,
             RESWK type ekko-RESWK,
           END OF LI_EKKO,
           LT_EKKO like table of li_ekko.

  Data  :  BEGIN OF LI_EKPO,
               EBELN type ekpo-ebeln,
               EBELP type ekpo-EBELP,
               MATNR type ekpo-MATNR,
               WERKS type ekpo-WERKS,
               MENGE type ekpo-MENGE,
               MEINS type ekpo-MEINS,
               UMREZ type ekpo-UMREZ,
               UMREN type ekpo-UMREN,
               NETWR type ekpo-NETWR,
               PSTYP type ekpo-PSTYP,
               RETPO type ekpo-RETPO,
               CCOMP type ekpo-CCOMP,
           END OF LI_EKPO,
          LT_EKPO like table of li_ekpo.

  Data  :  BEGIN OF LI_EKET,
                EBELN type eket-ebeln,
                EBELP type eket-EBELP,
                ETENR type eket-ETENR,
```

```
               WEMNG type eket-WEMNG,
               WAMNG type eket-WAMNG,
          END OF LI_EKET,
          LT_EKET like table of li_eket.



Data  :  BEGIN OF LI_EKPO_AGG,
               EBELN type eket-ebeln,
               EBELP type eket-EBELP,
               WEMNG type eket-WEMNG,
               WAMNG type eket-WAMNG,
               TRANSIT type eket-WAMNG,
          END OF LI_EKPO_AGG,
          LT_EKPO_AGG like table of li_ekpo_agg.

Data  :  BEGIN OF LI_EKBE,
                EBELN type ekbe-ebeln,
                EBELP type ekbe-EBELP,
                ZEKKN type ekbe-ZEKKN,
                VGABE type ekbe-VGABE,
                GJAHR type ekbe-GJAHR,
                BELNR type ekbe-BELNR,
                BUZEI type ekbe-BUZEI,
                BUDAT type ekbe-BUDAT,
                MENGE type ekbe-MENGE,
                SHKZG type ekbe-SHKZG,
           END OF LI_EKBE,
           LT_EKBE like table of li_ekbe.

Data : Begin of li_mara,
       matnr type mara-matnr,
       end of li_mara,
       lt_mara like table of li_mara.

Data : Begin of li_mara1,
       matnr type mara-matnr,
       meins type mara-meins,
       end of li_mara1,
       lt_mara1 like table of li_mara1.

Data : Loopx like sy-tabix.

Data : Begin of ls_T001,
       bukrs type T001-bukrs,
       waers type T001-waers,
       end of ls_t001,
       lt_t001 like table of ls_t001.


DATA: lt_data TYPE standard TABLE OF ZSTOCK_TRANSIT_1.

DATA: ls_data TYPE ZSTOCK_TRANSIT_1.

DATA: lv_lines TYPE sytfill,
      lv_maxsize TYPE rsmaxsize.

STATICS: lv_datapakid_counter TYPE sytabix,
         lv_cursor TYPE cursor,
         lv_tabix TYPE sytabix,
         lv_wamng TYPE menge_d,
         lv_wemng TYPE menge_d,
         lv_relevant TYPE char1.

DATA: lv_test TYPE char1 VALUE 'X'.
```

```
  DATA: lv_ebeln_field TYPE fieldname VALUE 'EBELN',
        lv_ebelp_field TYPE fieldname VALUE 'EBELP',
        lv_budat TYPE datum,
        lv_etenr_field TYPE fieldname VALUE 'ETENR'.


  DATA: lp_data TYPE REF TO data.

  FIELD-SYMBOLS: <ls_budat> TYPE ANY,
                 <lv_low> TYPE ANY.

  CREATE DATA lp_data LIKE LINE OF ir_budat.

  ASSIGN lp_data->* TO <ls_budat>.
  ASSIGN COMPONENT 'LOW' OF STRUCTURE <ls_budat> TO <lv_low>.

  READ TABLE ir_budat INTO <ls_budat> INDEX 1.
  lv_budat = <lv_low>.

  IF lv_datapakid_counter IS INITIAL.

    IF ir_ebeln is initial and ir_bsart is initial.
      MESSAGE e114(zstocktransit)
          RAISING error_passed_to_mess_handler.

*Parameter Purchase document number EBELN or Document type BSART is mandatory
    ENDIF.

** Select records from EKKO
    if not ir_ebeln is initial.

      OPEN CURSOR WITH HOLD lv_cursor FOR
          SELECT EBELN BUKRS BSTYP BSART BSAKZ WAERS WKURS KUFIX RESWK
          FROM ekko
          WHERE ebeln IN ir_ebeln.
    else.
      if not ir_bsart is initial.
        OPEN CURSOR WITH HOLD lv_cursor FOR
        SELECT EBELN BUKRS BSTYP BSART BSAKZ WAERS WKURS KUFIX RESWK
        FROM ekko
        WHERE bsart IN ir_bsart.

      endif.
    endif.

    IF sy-subrc NE 0.
      MESSAGE e103(zstocktransit) RAISING no_more_data.
* No data found
    ENDIF.
  ENDIF.

  FETCH NEXT CURSOR lv_cursor
    INTO CORRESPONDING FIELDS
    OF TABLE lt_ekko
    PACKAGE SIZE iv_maxsize.

  IF sy-subrc <> 0.
    CLOSE CURSOR lv_cursor.
    MESSAGE e103(zstocktransit) RAISING no_more_data.
  ENDIF.

  lv_datapakid_counter = lv_datapakid_counter + 1.

***Delete records from ekko where sending plant is blank and Category NE 'F'.
```

```
  delete lt_ekko where reswk eq space.

  delete lt_ekko where bstyp NE 'F'.

  delete lt_ekko where bsakz eq 'T'.

  sort lt_ekko by ebeln.

  if not lt_ekko is initial.

*** Select Company code currency

    SELECT BUKRS WAERS FROM t001 INTO TABLE lt_t001.

    sort lt_t001 by bukrs.

*** Select from EKPO

    SELECT EBELN EBELP MATNR WERKS MENGE MEINS
           UMREZ UMREN NETWR PSTYP RETPO CCOMP
    FROM ekpo INTO TABLE lt_ekpo FOR ALL ENTRIES IN lt_ekko
          WHERE ebeln EQ lt_ekko-ebeln.

*** Delete records from EKPO where item category NE '037' and NE plant given in the
selection.

    delete lt_ekpo where pstyp NA '037'.

    delete lt_ekpo where werks not in ir_werks.

    delete lt_ekpo where ccomp eq '1'.

    sort lt_ekpo by ebeln ebelp.

    if not lt_ekpo is initial.

*** If posting date in the selection is initial, select data from EKET, else from EKBE

      if lv_budat is initial.

        select ebeln ebelp etenr wemng wamng
        from eket into table lt_eket for all entries in lt_ekpo
        where ebeln eq lt_ekpo-ebeln
          and ebelp eq lt_ekpo-ebelp.

**** Aggregate EKET data from schedule line level to item level.

        loop at lt_eket into li_eket.
          move-corresponding li_eket to li_ekpo_agg.
          li_ekpo_agg-transit = li_eket-WAMNG - li_eket-wemng.
          collect li_ekpo_agg into lt_ekpo_agg.
        endloop.

*** Select from EKBE, if posting date is initial.

      else.
        select EBELN EBELP ZEKKN VGABE GJAHR BELNR BUZEI BUDAT MENGE SHKZG
          from EKBE into table lt_ekbe for all entries in lt_ekpo
             where ebeln eq lt_ekpo-ebeln
               and ebelp eq lt_ekpo-ebelp.

*** Delete records from EKBE, where event types are not linked to Goods receipt or Goods
Issue for stock transfer.
```

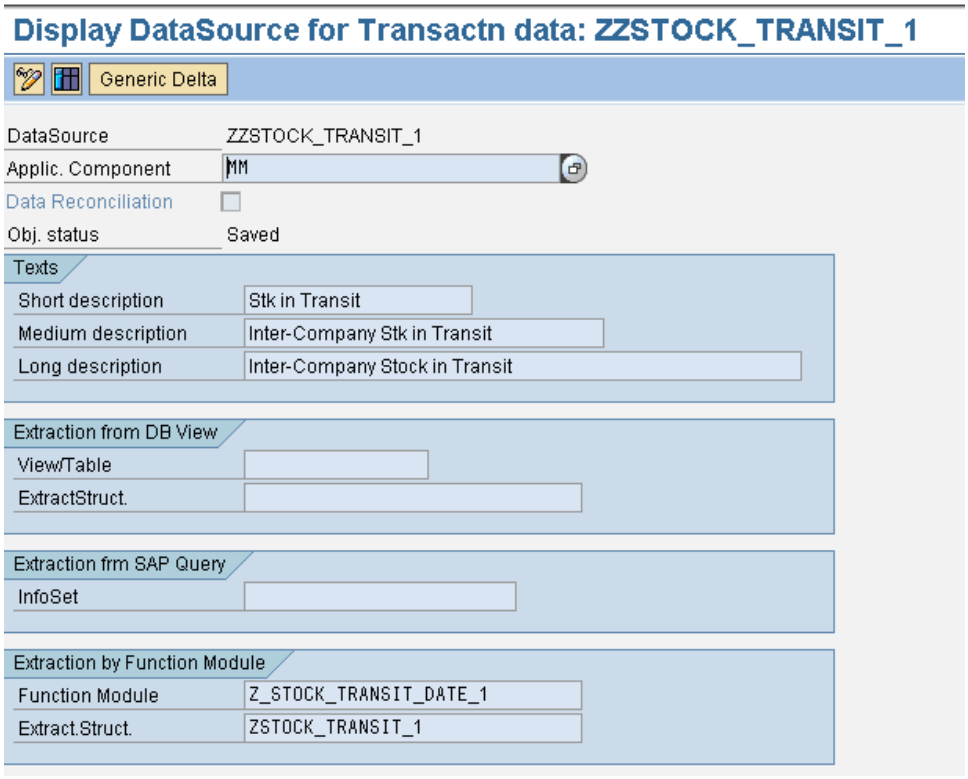
```
        delete lt_ekbe where ( vgabe ne '1' ) and
                             ( vgabe ne '6' ).

*** Delete records from EKBE where posting date is greater than the selection.
        delete lt_ekbe where budat > lv_budat.

        sort lt_ekbe by ebeln ebelp.
*** Aggregate EKBE data to Purchase order item level

        if not lt_ekbe is initial.
          loop at lt_ekbe into li_ekbe.
            clear : li_ekpo_agg.
            move-corresponding li_ekbe to li_ekpo_agg.
            CASE li_ekbe-vgabe.

              WHEN '1'.
*           goods receipt
                   IF li_ekbe-shkzg EQ 'S'.
*           normal goods receipt
                     li_ekpo_agg-wemng = li_ekbe-menge.
                   ELSE.
*           reversal
                     li_ekpo_agg-wemng = li_ekbe-menge * -1.
                   ENDIF.

              WHEN '6'.
*           goods issue
                   IF li_ekbe-shkzg EQ 'H'.
*           normal goods issue
                     li_ekpo_agg-wamng = li_ekbe-menge.
                   ELSE.
*           reversal
                     li_ekpo_agg-wamng = li_ekbe-menge * -1.

                   ENDIF.
            ENDCASE.

            li_ekpo_agg-transit = li_ekpo_agg-wamng - li_ekpo_agg-wemng.
            collect li_ekpo_agg into lt_ekpo_agg.
          endloop.
        endif.
      endif.

*** Delete records where there is no stock in transit quantity

      sort lt_ekpo_agg by ebeln ebelp.
      delete lt_ekpo_agg where transit eq '0'.

*** Populating final data

      loop at lt_ekpo_agg into li_ekpo_agg.

        Clear : ls_data.

        read table lt_ekko into li_ekko
               with key ebeln = li_ekpo_agg-ebeln binary search.

        if sy-subrc eq 0.
          read table lt_ekpo into li_ekpo
                 with key ebeln = li_ekpo_agg-ebeln
                          ebelp = li_ekpo_agg-ebelp binary search.
          if sy-subrc eq 0.
              move-corresponding li_ekko to ls_data.
              move-corresponding li_ekpo to ls_data.
              ls_data-menge = li_ekpo_agg-transit.
```

```
*** If posting date is initial, then use current date of currency conversion
              if lv_budat is initial.
                ls_data-budat = sy-datum.
              else.
                ls_data-budat = lv_budat.
              endif.
*** If Returns PO, reverse the quantity and populate sending plant in plant

              IF NOT li_ekpo-retpo IS INITIAL.
                ls_data-werks = li_ekko-reswk.
                IF ls_data-menge < 0.
                  ls_data-menge = ls_data-menge * -1.
                ENDIF.
              ENDIF.

* Find out PO rate by dividing PO value by quantity and multiply this rate with transit
quantity to get the Stock in Transit value

              ls_data-dmbtr = ls_data-menge * li_ekpo-netwr / li_ekpo-menge.
* Conversion to get the PO quantity to base unit of measurement, if the PO unit is
different.
              ls_data-menge = ls_data-menge * li_ekpo-umrez / li_ekpo-umren.
              READ TABLE lt_t001 INTO ls_t001 WITH KEY bukrs = ls_data-bukrs BINARY
SEARCH.
              if sy-subrc eq 0.

* If company code currency is not same as PO currency, convert to company code currency

                IF ls_t001-waers NE li_ekko-waers.

*Check for Fixed Currency Rate indicator in PO - NM 18082008
                  IF li_ekko-kufix is initial.
                    CALL FUNCTION 'CONVERT_TO_LOCAL_CURRENCY'
                    EXPORTING
                      date             = ls_data-budat
                      foreign_amount   = ls_data-dmbtr
                      foreign_currency = li_ekko-waers
                      local_currency   = ls_t001-waers

*Commented by NM 12082008 to use current exchange instead of rate in PO document
*                      rate = li_ekko-wkurs
                    IMPORTING
                      local_amount   = ls_data-dmbtr.
                      ls_data-hwaer  = ls_t001-waers.
                  ELSEIF li_ekko-kufix = 'X'.
                    CALL FUNCTION 'CONVERT_TO_LOCAL_CURRENCY'
                      EXPORTING
                        date             = ls_data-budat
                        foreign_amount   = ls_data-dmbtr
                        foreign_currency = li_ekko-waers
                        local_currency   = ls_t001-waers
                        rate             = li_ekko-wkurs
                      IMPORTING
                        local_amount     = ls_data-dmbtr.
                        ls_data-hwaer    = ls_t001-waers.
                  ENDIF.

                ELSE.
                  ls_data-hwaer = li_ekko-waers.
                ENDIF.
              endif.
** Populate internal table to get unique materials to select from MARA
              li_mara-matnr = ls_data-matnr.
              append li_mara to lt_mara.
*** Append final internal table
```

```
              append ls_data to lt_data.
          endif.
        endif.
      endloop.
*** get unique materials.
      sort lt_mara by matnr.
      delete adjacent duplicates from lt_mara.
*** Select base unit of measurement from MARA
      if not lt_mara is initial.
        Select matnr meins from mara into table lt_mara1
             for all entries in lt_mara
               where matnr = lt_mara-matnr.
        sort lt_mara1 by matnr.
        loop at lt_data into ls_data.
          loopx = sy-tabix.
          read table lt_mara1 into li_mara1 with key
                       matnr = ls_data-matnr binary search.
          if sy-subrc eq 0.
            ls_data-meins = li_mara1-meins.
            modify lt_data from ls_data index loopx transporting meins.
          endif.
        endloop.
      endif.
    endif.
  endif.
  ct_data = lt_data.
  Refresh : lt_ekko, lt_ekpo, lt_ekbe, lt_eket, lt_ekpo_agg, lt_mara, lt_mara1, lt_data,
lt_t001.

ENDFORM. " data_selection_transit
*&---------------------------------------------------------------------*
*&      Form  range_fill
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->IV_SELFIELD  text
*      -->IT_SELTAB    text
*      -->ER_RANGE     text
*----------------------------------------------------------------------*
FORM range_fill USING iv_selfield
                      it_seltab TYPE srsc_t_select
                CHANGING er_range TYPE STANDARD TABLE.

  DATA: lp_data TYPE REF TO data.
  DATA: ls_seltab TYPE srsc_s_select.
  FIELD-SYMBOLS: <ls_line> TYPE ANY.

  CREATE DATA lp_data LIKE LINE OF er_range.
  ASSIGN lp_data->* TO <ls_line>.

  LOOP AT it_seltab INTO ls_seltab
       WHERE fieldnm = iv_selfield.

    PERFORM build_range USING ls_seltab-sign
                              ls_seltab-option
                              ls_seltab-low
                              ls_seltab-high
                      CHANGING er_range.
  ENDLOOP.
ENDFORM. " range_fill
```

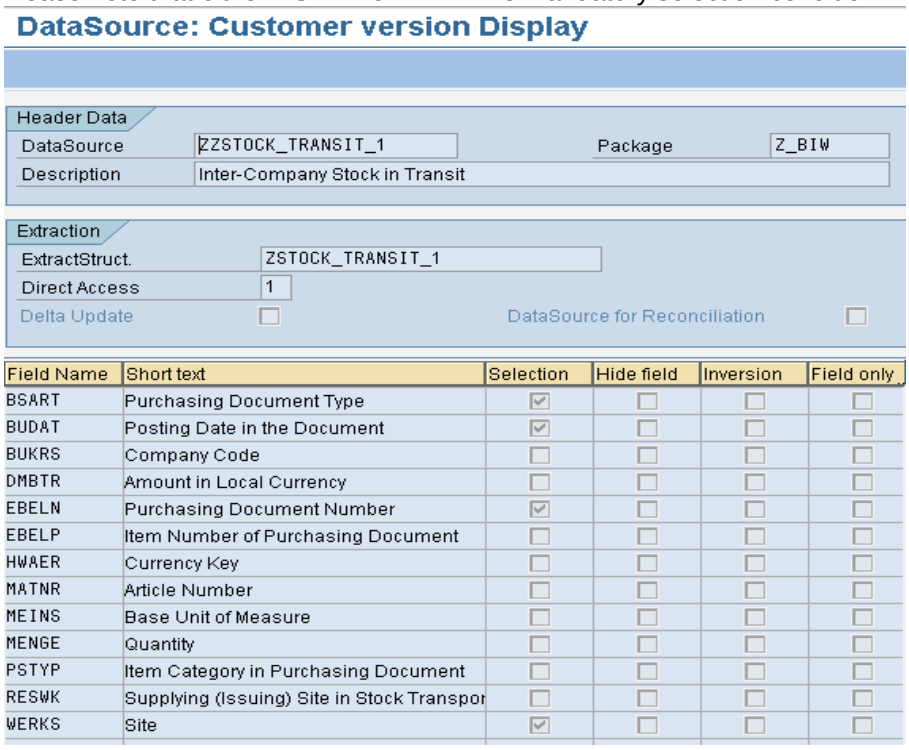
3) Creation of Datasource

Create the datasource ZZSTOCK_TRANSIT_1 in Transaction RSO2 using the above function module and extract structure.



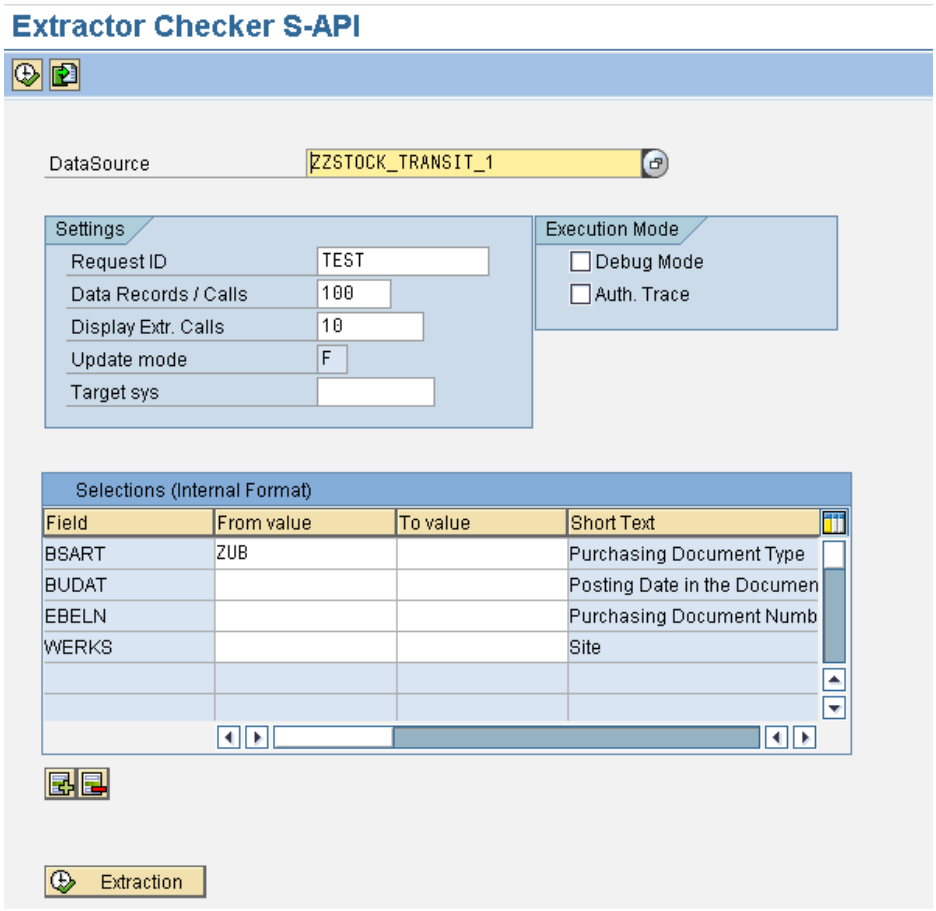
Enable the fields for giving selection inputs as shown below..

Please note that either BSART or EBELN is mandatory selection condition.



Field Name	Short text	Selection	Hide field	Inversion	Field only
BSART	Purchasing Document Type	☑	☐	☐	☐
BUDAT	Posting Date in the Document	☑	☐	☐	☐
BUKRS	Company Code	☐	☐	☐	☐
DMBTR	Amount in Local Currency	☐	☐	☐	☐
EBELN	Purchasing Document Number	☑	☐	☐	☐
EBELP	Item Number of Purchasing Document	☐	☐	☐	☐
HWAER	Currency Key	☐	☐	☐	☐
MATNR	Article Number	☐	☐	☐	☐
MEINS	Base Unit of Measure	☐	☐	☐	☐
MENGE	Quantity	☐	☐	☐	☐
PSTYP	Item Category in Purchasing Document	☐	☐	☐	☐
RESWK	Supplying (Issuing) Site in Stock Transpor	☐	☐	☐	☐
WERKS	Site	☑	☐	☐	☐

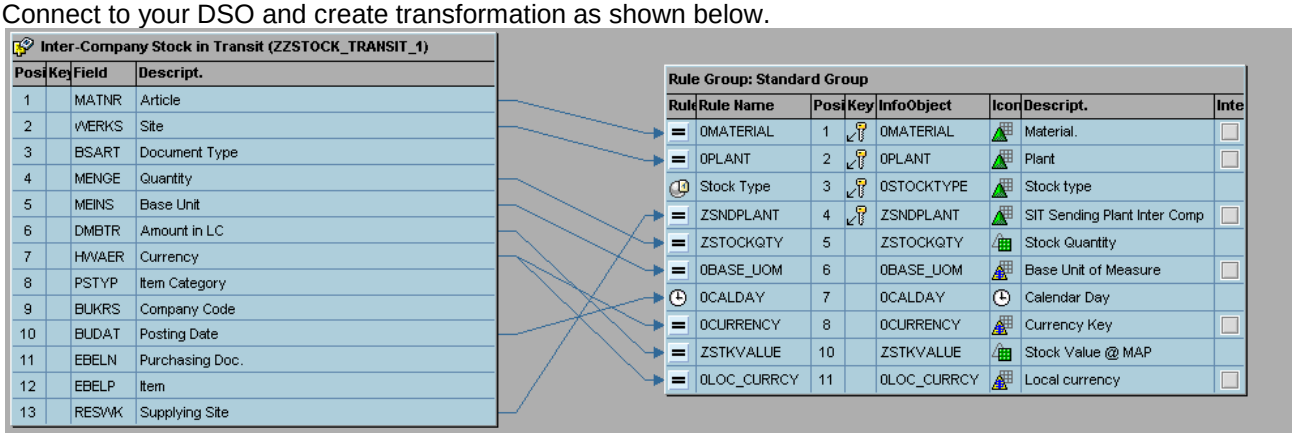
Check the extractor in RSA3.


Extractor Checker S-API
DataSource
ZZSTOCK_TRANSIT_1
Settings
Request ID
TEST
Data Records / Calls
100
Display Extr. Calls
10
Update mode
F
Target sys
Execution Mode
Debug Mode
Auth. Trace
Selections (Internal Format)
Field
From value
To value
Short Text
BSART
ZUB
Purchasing Document Type
BUDAT
Posting Date in the Documen
EBELN
Purchasing Document Numb
WERKS
Site
Extraction

Steps in BW

1) Replicate the datasource in BW.

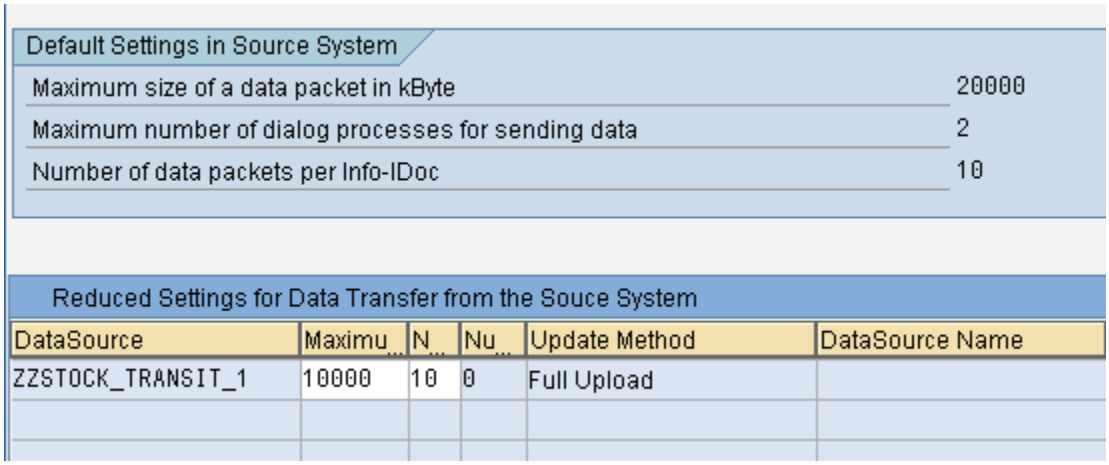
2) Connect to your DSO and create transformation as shown below.



Please note that the keyfigures should be in addition mode. Alternatively, you can keep Purchase document number and Line item in the DSO keys and rest of the infoobjects in the data fields. This way the keyfigures can be kept in 'Overwrite' mode. This DSO has to be dropped every time and should be loaded in full update mode from the source.

3) You can connect this DSO to a Cube and do a full upload to take daily or weekly snapshots. A multiprovider can be created which can combine this SIT Cube with Stock cube to give complete stock position.

4) For better performance and to limit the number of purchase order items processed in each package, it is better to limit the maximum size of the data packet to 10000 or 5000 (Default 20000) in the infopackage -> Scheduler -> Datas. Default data transfer as shown below.



5) You can improve the performance further by creating multiple infopackages with each filtered on different purchase order document types those are relevant for cross company stock in transit and by scheduling those infopackages in parallel.

Related Content

How to Report on Cross Company Stock in Transit

How to Handle Inventory Management Scenarios in BW (NW2004)

Non-Cumulatives - Stock Handling

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.