PPRK904571

TABLE : EKPO

Variante de selection : ZTEST1 -> gt\_data

MISE EN FORME: ZTEST -> Col1 COl2

TRie par col

champ =!

**Variante Selection**

DATA : BEGIN OF itab OCCURS 0,  
         matnr LIKE mara-matnr,  
       END OF itab.  
  
TYPES: BEGIN OF ty\_s\_clause.  
TYPES:   line(72)  TYPE c.  
TYPES: END OF ty\_s\_clause.  
  
DATA : BEGIN OF gt\_condtab OCCURS 0.  
         INCLUDE STRUCTURE hrcond.  
       DATA : END   OF gt\_condtab.  
  
FIELD-SYMBOLS <fs\_wherecond> TYPE ty\_s\_clause.  
DATA:  
  gt\_where\_clauses  TYPE STANDARD TABLE OF ty\_s\_clause  
                    WITH DEFAULT KEY.  
  
DATA: lt\_variant   TYPE TABLE OF se16n\_lt,  
      lt\_selection TYPE TABLE OF se16n\_ltd.  
  
SELECT \*  
FROM se16n\_lt  
  INTO TABLE @lt\_variant  
  WHERE name EQ 'ZMDLIMI'  
    AND tab  EQ 'BKPF'.  
IF sy-subrc EQ 0.  
  
  SELECT \*  
   FROM se16n\_ltd  
   INTO TABLE @lt\_selection  
    FOR ALL ENTRIES IN @lt\_variant  
   WHERE guid    EQ @lt\_variant-guid  
     AND counter EQ '001'.  
  IF sy-subrc EQ 0.  
    LOOP AT lt\_selection INTO DATA(ls\_selection).  
      gt\_condtab-field = ls\_selection-field.  
      IF ls\_selection-sign EQ 'I'.  
        IF ls\_selection-high IS INITIAL.  
          gt\_condtab-opera = 'EQ'.  
        ELSE.  
          gt\_condtab-opera = 'IN'.  
        ENDIF.  
      ELSEIF ls\_selection-optio IS NOT INITIAL.  
        gt\_condtab-opera = ls\_selection-optio.  
      ENDIF.  
      gt\_condtab-low   = ls\_selection-low.  
      gt\_condtab-high  = ls\_selection-high.  
      APPEND  gt\_condtab.  
      CLEAR  gt\_condtab.  
    ENDLOOP.  
  ENDIF.  
  
ENDIF.  
  
CALL FUNCTION 'RH\_DYNAMIC\_WHERE\_BUILD'  
  EXPORTING  
    dbtable         = space *" can be empty*  
  TABLES  
    condtab         = gt\_condtab  
    where\_clause    = gt\_where\_clauses  
  EXCEPTIONS  
    empty\_condtab   = 01  
    no\_db\_field     = 02  
    unknown\_db      = 03  
    wrong\_condition = 04.  
.  
DATA: l\_table\_name(10) TYPE c.  
  
l\_table\_name = 'BKPF'.  
  
DATA: dy\_table TYPE REF TO data.  
FIELD-SYMBOLS: <dyn\_table> TYPE STANDARD TABLE.  
CREATE DATA dy\_table TYPE STANDARD TABLE OF (l\_table\_name).  
ASSIGN dy\_table->\* TO <dyn\_table>.  
  
SELECT \* FROM (l\_table\_name)  
  INTO CORRESPONDING FIELDS OF TABLE <dyn\_table>  
   WHERE (gt\_where\_clauses).

**Variante de mise en forme**

***"Get fields of Variante Layout*  
DATA: wa\_ltdx        TYPE ltdx,  
      v\_varkey       TYPE ltdxkey,  
      it\_fcat        TYPE STANDARD TABLE OF ltdxdata,  
      it\_sort\_info   TYPE STANDARD TABLE OF ltdxdata,  
      ls\_sort        TYPE ltdxdata,  
      lt\_sort        TYPE STANDARD TABLE OF ltdxdata,  
      it\_filter      TYPE STANDARD TABLE OF ltdxdata,  
      lt\_zones\_types TYPE STANDARD TABLE OF dd03l,  
      lr\_pay\_data    TYPE REF TO data,  
      ls\_metadata    TYPE cl\_salv\_bs\_runtime\_info=>s\_type\_metadata,  
      t\_filters      TYPE TABLE OF ty\_filter,  
      s\_filters      TYPE ty\_filter,  
      l\_index        TYPE i.  
  
CONSTANTS: c\_relid          TYPE ltdx-relid VALUE 'LT',  
           c\_structure\_me2n TYPE dd03l-tabname VALUE 'MEREP\_OUTTAB\_PURCHDOC',  
           c\_prog           TYPE ltdx-report VALUE 'RM06EN00',  
           c\_varname        TYPE ltdx-variant VALUE '/IMPORT/VDOC'.  
  
SELECT SINGLE \* FROM ltdx INTO wa\_ltdx  
WHERE relid   = c\_relid  
  AND report  = c\_prog  
  AND variant = c\_varname.  
  
IF sy-subrc = 0.  
  
  MOVE-CORRESPONDING wa\_ltdx TO v\_varkey.  
*\* will get the layout fields*  
  
  CALL FUNCTION 'LT\_DBDATA\_READ\_FROM\_LTDX'  
    EXPORTING  
      i\_tool       = c\_relid  
      is\_varkey    = v\_varkey  
    TABLES  
      t\_dbfieldcat = it\_fcat      *" To create Field Catalog*  
      t\_dbsortinfo = it\_sort\_info *" Sort category set*  
      t\_dbfilter   = it\_filter.   *" Filter category set*  
  
  DELETE it\_fcat WHERE param NE 'NO\_OUT'.  
  DELETE it\_fcat WHERE value EQ 'X'.  
  
  SORT it\_filter BY key1.  
  LOOP AT it\_filter INTO DATA(ls\_filter).  
  
    AT NEW key1.  
      ADD 1 TO l\_index.  
    ENDAT.  
  
    s\_filters-index = l\_index.  
    s\_filters-fieldname = ls\_filter-key1.  
    IF ls\_filter-param EQ 'SIGN0'.  
      s\_filters-sign = ls\_filter-value.  
    ENDIF.  
    IF ls\_filter-param EQ 'OPTIO'.  
      s\_filters-option = ls\_filter-value.  
    ENDIF.  
    IF ls\_filter-param EQ 'VALUF\_INT'.  
      s\_filters-low = ls\_filter-value.  
    ENDIF.  
  
    IF s\_filters-sign   IS NOT INITIAL AND  
       s\_filters-option IS NOT INITIAL AND  
       s\_filters-low    IS NOT INITIAL.  
  
      APPEND s\_filters TO t\_filters.  
      CLEAR s\_filters.  
  
    ENDIF.  
  
  ENDLOOP.  
  
ENDIF.**

**FIELD-SYMBOLS: <fs\_any>.**

**DATA: l\_check\_true TYPE abap\_bool.  
  
DESCRIBE TABLE t\_filters LINES DATA(lines).  
READ TABLE t\_filters INTO DATA(last\_line) INDEX lines.  
  
CLEAR l\_index.  
LOOP AT <lt\_pay\_data> ASSIGNING <ls\_pay\_data>.  
  
  DO last\_line-index TIMES.  
    ADD 1 TO l\_index.  
    READ TABLE t\_filters INTO DATA(l\_filter) WITH KEY index = l\_index.  
    ASSIGN COMPONENT l\_filter-fieldname OF STRUCTURE <ls\_pay\_data> TO <fs\_any>.  
    LOOP AT t\_filters INTO s\_filters WHERE index = l\_index.  
      CASE s\_filters-option.  
        WHEN 'EQ'.  
          CASE s\_filters-sign.  
            WHEN 'I'.  
              IF <fs\_any> = s\_filters-low.  
                l\_check\_true = abap\_true.  
                EXIT.  
              ELSE.  
                l\_check\_true = abap\_false.  
              ENDIF.  
            WHEN 'E'.  
              IF <fs\_any> NE s\_filters-low.  
                l\_check\_true = abap\_true.  
              ELSE.  
                l\_check\_true = abap\_false.  
                EXIT.  
              ENDIF.  
          ENDCASE.  
        WHEN 'NE'.  
          CASE s\_filters-sign.  
            WHEN 'I'.  
              IF <fs\_any> NE s\_filters-low.  
                l\_check\_true = abap\_true.  
                EXIT.  
              ELSE.  
                l\_check\_true = abap\_false.  
              ENDIF.  
            WHEN 'E'.  
              IF <fs\_any> EQ s\_filters-low.  
                l\_check\_true = abap\_true.  
              ELSE.  
                l\_check\_true = abap\_false.  
                EXIT.  
              ENDIF.  
          ENDCASE.  
        WHEN 'GE'.  
          CASE s\_filters-sign.  
            WHEN 'I'.  
              IF <fs\_any> GE s\_filters-low.  
                l\_check\_true = abap\_true.  
                EXIT.  
              ELSE.  
                l\_check\_true = abap\_false.  
              ENDIF.  
            WHEN 'E'.  
              IF <fs\_any> LT s\_filters-low.  
                l\_check\_true = abap\_true.  
              ELSE.  
                l\_check\_true = abap\_false.  
                EXIT.  
              ENDIF.  
          ENDCASE.  
        WHEN 'GT'.  
          CASE s\_filters-sign.  
            WHEN 'I'.  
              IF <fs\_any> GT s\_filters-low.  
                l\_check\_true = abap\_true.  
                EXIT.  
              ELSE.  
                l\_check\_true = abap\_false.  
              ENDIF.  
            WHEN 'E'.  
              IF <fs\_any> LE s\_filters-low.  
                l\_check\_true = abap\_true.  
              ELSE.  
                l\_check\_true = abap\_false.  
                EXIT.  
              ENDIF.  
          ENDCASE.  
        WHEN 'LE'.  
          CASE s\_filters-sign.  
            WHEN 'I'.  
              IF <fs\_any> LE s\_filters-low.  
                l\_check\_true = abap\_true.  
                EXIT.  
              ELSE.  
                l\_check\_true = abap\_false.  
              ENDIF.  
            WHEN 'E'.  
              IF <fs\_any> GT s\_filters-low.  
                l\_check\_true = abap\_true.  
              ELSE.  
                l\_check\_true = abap\_false.  
                EXIT.  
              ENDIF.  
          ENDCASE.  
        WHEN 'LT'.  
          CASE s\_filters-sign.  
            WHEN 'I'.  
              IF <fs\_any> LT s\_filters-low.  
                l\_check\_true = abap\_true.  
                EXIT.  
              ELSE.  
                l\_check\_true = abap\_false.  
              ENDIF.  
            WHEN 'E'.  
              IF <fs\_any> GE s\_filters-low.  
                l\_check\_true = abap\_true.  
              ELSE.  
                l\_check\_true = abap\_false.  
                EXIT.  
              ENDIF.  
          ENDCASE.  
        WHEN 'CP'.  
          CASE s\_filters-sign.  
            WHEN 'I'.  
              IF <fs\_any> CP s\_filters-low.  
                l\_check\_true = abap\_true.  
                EXIT.  
              ELSE.  
                l\_check\_true = abap\_false.  
              ENDIF.  
            WHEN 'E'.  
              IF <fs\_any> NP s\_filters-low.  
                l\_check\_true = abap\_true.  
              ELSE.  
                l\_check\_true = abap\_false.  
                EXIT.  
              ENDIF.  
          ENDCASE.  
        WHEN 'NP'.  
          CASE s\_filters-sign.  
            WHEN 'I'.  
              IF <fs\_any> NP s\_filters-low.  
                l\_check\_true = abap\_true.  
                EXIT.  
              ELSE.  
                l\_check\_true = abap\_false.  
              ENDIF.  
            WHEN 'E'.  
              IF <fs\_any> CP s\_filters-low.  
                l\_check\_true = abap\_true.  
              ELSE.  
                l\_check\_true = abap\_false.  
                EXIT.  
              ENDIF.  
          ENDCASE.  
        WHEN 'BT'.  
          CASE s\_filters-sign.  
            WHEN 'I'.  
              IF ( <fs\_any> GE s\_filters-low AND <fs\_any> LE s\_filters-high ).  
                l\_check\_true = abap\_true.  
                EXIT.  
              ELSE.  
                l\_check\_true = abap\_false.  
              ENDIF.  
            WHEN 'E'.  
              IF ( <fs\_any> LT s\_filters-low AND <fs\_any> GT s\_filters-high ).  
                l\_check\_true = abap\_true.  
              ELSE.  
                l\_check\_true = abap\_false.  
                EXIT.  
              ENDIF.  
          ENDCASE.  
      ENDCASE.  
    ENDLOOP.  
    IF l\_check\_true EQ abap\_false.  
      EXIT.  
    ENDIF.  
  ENDDO.  
  
  IF l\_check\_true EQ abap\_false.  
    CLEAR l\_index.  
    CONTINUE.  
  ENDIF.  
  CLEAR l\_index.  
  
  IF l\_check\_true EQ abap\_true.  
    MOVE-CORRESPONDING <ls\_pay\_data> TO <fs\_wa>.  
    APPEND <fs\_wa> TO <fs\_table>.  
    CLEAR l\_check\_true.  
  ENDIF.  
  
ENDLOOP.  
  
*\*Sort Table*  
DATA: t\_fields TYPE abap\_sortorder\_tab,  
      ls\_field TYPE abap\_sortorder.  
  
IF <fs\_table> IS NOT INITIAL AND it\_sort\_info[] IS NOT INITIAL.  
  
  LOOP AT it\_sort\_info INTO DATA(l\_sort).  
    CASE l\_sort-param.  
      WHEN 'UP'.  
        CHECK l\_sort-value EQ 'X'.  
        ls\_field-name = l\_sort-key1.  
        APPEND ls\_field TO t\_fields.  
        CLEAR ls\_field.  
      WHEN 'DOWN'.  
        CHECK l\_sort-value EQ 'X'.  
        ls\_field-name       = l\_sort-key1.  
        ls\_field-descending = 'X'.  
        APPEND ls\_field TO t\_fields.  
        CLEAR ls\_field.  
    ENDCASE.  
  ENDLOOP.  
  
  SORT <fs\_table> BY (t\_fields).  
  
ENDIF.**

**Fomattage date  
        DESCRIBE TABLE it\_fcat LINES DATA(l\_taille).  
        LOOP AT lt\_data\_output INTO DATA(ls\_data\_output).  
          LOOP AT it\_fcat INTO ls\_fcat.  
            ASSIGN COMPONENT ls\_fcat-key1 OF STRUCTURE ls\_data\_output TO <fs>.  
            l\_value = <fs>.  
            SHIFT l\_value LEFT DELETING LEADING space.  
            IF ls\_fcat-key1 EQ 'AUGDT' OR  
               ls\_fcat-key1 EQ 'BLDAT' OR  
               ls\_fcat-key1 EQ 'BUDAT' OR  
               ls\_fcat-key1 EQ 'FAEDT' OR  
               ls\_fcat-key1 EQ 'MADAT' OR  
               ls\_fcat-key1 EQ 'VALUT' OR  
               ls\_fcat-key1 EQ 'ZALDT' OR  
               ls\_fcat-key1 EQ 'ZFBDT' OR  
               ls\_fcat-key1 EQ '/SAPF15/ERSDA' OR  
               ls\_fcat-key1 EQ '/SAPF15/LAEDA' OR  
               ls\_fcat-key1 EQ 'U\_CPUDT'.  
  
              l\_value = l\_value(4) && '-' && l\_value+4(2) && '-' && l\_value+6(2).  
  
            ELSEIF l\_value CS '-'.  
              REPLACE '-' INTO l\_value WITH space.  
              CONCATENATE '-' l\_value INTO l\_value.  
              CONDENSE l\_value NO-GAPS.  
            ENDIF.  
            IF NOT sy-tabix EQ l\_taille.  
              CONCATENATE l\_value ';' INTO l\_value.  
              CONCATENATE line\_value l\_value INTO line\_value.  
            ELSE.  
              CONCATENATE line\_value l\_value INTO line\_value.  
            ENDIF.  
          ENDLOOP.  
          TRANSFER line\_value TO lv\_filename.  
          CLEAR line\_value.  
        ENDLOOP.**

**Elements de données**

**Table: DD03L**

**filters**

**LOOP AT it\_filter INTO DATA(ls\_filter).**

**AT NEW key1.**

**ADD 1 TO l\_index.**

**ENDAT.**

**s\_filters-index = l\_index.**

**s\_filters-fieldname = ls\_filter-key1.**

**IF ls\_filter-param EQ 'SIGN0'.**

**s\_filters-sign = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'SIGN\_ICON'.**

**s\_filters-sign\_icon = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'OPTIO'.**

**s\_filters-option = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'VALUF\_INT'.**

**s\_filters-low = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'VALUT\_INT'.**

**s\_filters-high = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'VALUF'.**

**s\_filters-low\_ext = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'VALUT'.**

**s\_filters-high\_ext = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'STYPE'.**

**s\_filters-selection\_type = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'EXCEPTION'.**

**s\_filters-exception = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'OR'.**

**s\_filters-or\_group = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'CQVALUE'.**

**s\_filters-critical\_value = ls\_filter-value.**

**ENDIF.**

**IF ls\_filter-param EQ 'DECIMALS'.**

**s\_filters-decimals = ls\_filter-value.**

**ENDIF.**

**IF s\_filters-sign IS NOT INITIAL AND**

**s\_filters-option IS NOT INITIAL AND**

**s\_filters-low IS NOT INITIAL.**

**APPEND s\_filters TO t\_filters.**

**CLEAR s\_filters.**

**ENDIF.**

**ENDLOOP.**