Working with Aggregate Functions for ABAP

Applies to:

SAP ECC 6.0 (Release 700, SP 12). For more information, visit the ABAP homepage.

Summary

This article explains how to use aggregate functions in ABAP.

Author: Rahul Babukuttan

Company: Applexus Software Solutions Ltd.

Created on: 11 November 2010

Author Bio

Rahul Babukuttan, Applexus Software Solutions Ltd.

B.Tech Computer Science Graduate, working as SAP Technology Consultant.

Table of Contents

What are Aggregate Functions?

Aggregate functions are functions that return a single value from the values in several lines in a column in a database table. The value is calculated in the database system. The specification of a column in a database table as an argument for an aggregate function constitutes an aggregate expression, which can be used in the SELECT statement.

The different aggregate functions are

- Average : AVG
- Minimum : MIN
- Maximum : MAX
- Total : SUM
- Count : COUNT

Average - AVG

Returns the average value in the column determined by the database field for the selected lines. AVG can only apply to a numeric field. The result is a NULL value only if all the rows in the column in question contain the NULL value. Otherwise NULL values are ignored. Retaining duplicates is important property while computing an average. If we want to eliminate duplicates, we use the keyword DISTINCT. Aggregate expressions with the function AVG have the data type FLTP.

```
DATA: lv_avgnetwr TYPE f.
PARAMETERS: p_vbeln TYPE vbeln_va.

  SELECT AVG( netwr ) FROM vbap
      INTO lv_avgnetwr
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Average value of Net-value is :', lv_avgnetwr.
  ENDIF.
```

Minimum - MIN

Returns the smallest value in the column determined by the database field for the selected lines. Specifying DISTINCT does not change the result. The result is a NULL value only if all the rows in the column in question contain the NULL value. Otherwise NULL values are ignored. The data type of aggregate expressions with the MIN is the data type of the corresponding column in the ABAP Dictionary.

```
DATA: lv_minkwmeng TYPE kwmeng.
PARAMETERS: p_vbeln TYPE vbeln_va.
  SELECT MIN( kwmeng ) FROM vbap
      INTO lv_minkwmeng
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Minimum Cumulative Order Quantity is :', lv_minkwmeng.
  ENDIF.
```

Maximum - MAX

Returns the greatest value in the column determined by the database field for the selected lines. Specifying DISTINCT does not change the result. The result is a NULL value only if all the rows in the column in question contain the NULL value. Otherwise NULL values are ignored. The data type of aggregate expressions with the MAX is the data type of the corresponding column in the ABAP Dictionary.

```
DATA: lv_maxkwmeng TYPE kwmeng.
PARAMETERS: p_vbeln TYPE vbeln_va.

  SELECT MAX( kwmeng ) FROM vbap
      INTO lv_maxkwmeng
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Maximum Cumulative Order Quantity is :', lv_maxkwmeng.
  ENDIF.
```

Total - SUM

Returns the sum of all values in the column determined by the database field for the selected lines. SUM can only apply to a numeric field. The result is a NULL value only if all the rows in the column in question contain the NULL value. Otherwise NULL values are ignored. In this case, the result is NULL. The data type of the column has to be numerical. For the aggregate function SUM, make the data type of the target field longer than the data type of the source field, to allow for overflow.

```
DATA: lv_sumkwmeng TYPE p decimals 2.
PARAMETERS: p_vbeln TYPE vbeln_va.
  SELECT SUM( kwmeng ) FROM vbap
      INTO lv_sumkwmeng
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Sum of Net-Value is:', lv_sumkwmeng.
  ENDIF.
```

Count – COUNT

COUNTDISTINCT returns the number of different values in the column determined by the database field for the selected lines. NULL values are ignored unless all values in a column are NULL values. In this case, the result is 0. COUNT (*) (or count(*)) determines the number of rows in the resulting set or in the current group. No column label is specified in this case. Aggregate expressions with the function COUNT have the data type INT4.

```
DATA: lv_countmatnr TYPE int4.
PARAMETERS: p_vbeln TYPE vbeln_va.
  SELECT COUNT( DISTINCT matnr ) FROM vbap
      INTO lv_countmatnr
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Total No of Distinct Matrials :', lv_countmatnr.
  ENDIF.
```

If you specify aggregate functions together with one or more database fields in a SELECT clause, all database fields not used in one of the aggregate functions must be listed in the GROUP-BY clause.

Note: Pooled and Cluster tables do not any support aggregate functions other than COUNT (*). Columns of the type STRING or RAWSTRING cannot be used with aggregate functions.

In Reports the use of aggregate functions can be illustrated using the following Abap code.

```
*&---------------------------------------------------------------------*
*& Report  Z_AGGREGATE
*&
*&---------------------------------------------------------------------*
*&
*&
*&
*&---------------------------------------------------------------------*

REPORT  z_aggregate.

DATA: t_vbap TYPE STANDARD TABLE OF vbap,
      x_vbap TYPE vbap.

SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME TITLE text-000.
PARAMETERS: p_vbeln TYPE vbeln_va MATCHCODE OBJECT zp_vbeln.
SELECTION-SCREEN END OF BLOCK b1.

START-OF-SELECTION.

  PERFORM fetch_counttotal."find total no of records for selection
  PERFORM fetch_maxkwmeng."find maximum cumulative order quantity
  PERFORM fetch_minkwmeng."find minimum cumulative order quantity
  PERFORM fetch_avgnetwr. "find average Netvalue
  PERFORM fetch_countmatnr."count distinct materials
  PERFORM fetch_sumkwmeng."find sum of net value

END-OF-SELECTION.
*&---------------------------------------------------------------------*
*&      Form  FETCH_AVGNETWR
*&---------------------------------------------------------------------*
*       "find average Netvalue
*----------------------------------------------------------------------*
FORM fetch_avgnetwr .

  DATA: lv_avgnetwr TYPE f.
  SELECT AVG( netwr ) FROM vbap
      INTO lv_avgnetwr
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Average value of Net-value is :', lv_avgnetwr.
  ENDIF.

ENDFORM.                    " FETCH_AVGNETWR
*&---------------------------------------------------------------------*
*&      Form  FETCH_MAXKWMENG
*&---------------------------------------------------------------------*
*       "find maximum cumulative order quantity
*----------------------------------------------------------------------*
FORM fetch_maxkwmeng .

  DATA: lv_maxkwmeng TYPE kwmeng.
  SELECT MAX( kwmeng ) FROM vbap
```

```
      INTO lv_maxkwmeng
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Maximum Cumulative Order Quantity is :', lv_maxkwmeng.
  ENDIF.

ENDFORM.                    " FETCH_MAXKWMENG
*&---------------------------------------------------------------------*
*&      Form  FETCH_MINKWMENG
*&---------------------------------------------------------------------*
*       "find minimum cumulative order quantity
*----------------------------------------------------------------------*
FORM fetch_minkwmeng .

  DATA: lv_minkwmeng TYPE kwmeng.
  SELECT MIN( kwmeng ) FROM vbap
      INTO lv_minkwmeng
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Minimum Cumulative Order Quantity is :', lv_minkwmeng.
  ENDIF.

ENDFORM.                    " FETCH_MINKWMENG
*&---------------------------------------------------------------------*
*&      Form  FETCH_COUNTTOTAL
*&---------------------------------------------------------------------*
*       "find total no of records for selection
*----------------------------------------------------------------------*
FORM fetch_counttotal .

  DATA: lv_counttotal TYPE i.
  SELECT COUNT( * ) FROM vbap
      INTO lv_counttotal
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Total No of records for selection criteria :', lv_counttotal.
  ENDIF.

ENDFORM.                    " FETCH_COUNTTOTAL
*&---------------------------------------------------------------------*
*&      Form  FETCH_COUNTMATNR
*&---------------------------------------------------------------------*
*      "count distinct materials
*----------------------------------------------------------------------*
FORM fetch_countmatnr .

  DATA: lv_countmatnr TYPE int4.
  SELECT COUNT( DISTINCT matnr ) FROM vbap
      INTO lv_countmatnr
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Total No of Distinct Matrials :', lv_countmatnr.
```

```
  ENDIF.

ENDFORM.                    " FETCH_COUNTMATNR
*&---------------------------------------------------------------------*
*&      Form  FETCH_SUMKWMENG
*&---------------------------------------------------------------------*
*       "find sum of net value
*---------------------------------------------------------------------*
form FETCH_SUMKWMENG .

  DATA: lv_sumkwmeng TYPE p DECIMALS 2.
  SELECT sum( kwmeng ) FROM vbap
      INTO lv_sumkwmeng
      WHERE vbeln EQ p_vbeln.

  IF sy-subrc EQ 0.
    WRITE: /'Sum of Net-Value is :', lv_sumkwmeng.
  ENDIF.

endform.                    " FETCH_SUMKWMENG
```

Advantages of using aggregate functions

Using aggregate functions significantly reduces the quantity of data records to be transferred. Overall Network, Application-server and Database load is also considerably less while using aggregate functions. Aggregate functions can also work across multiple fields. By adding a GROUP BY clause at the end of our select statement, we can group together fields to give a total value for each grouped field. Aggregate functions (SUM, MIN, MAX, AVG, etc.) are the preferred method for performing mathematical and statistical functions on database records.

Performing these functions programmatically would required transferring too much data from the database to the application. Aggregate functions limit the amount of data transferred between the database and the application.

Disadvantages of using aggregate functions

ABAP statements that contain aggregate functions bypass R/3 table buffers. This can be a disadvantage if tables are buffered. Pooled and Cluster tables do not any support aggregate functions other than COUNT (*). Columns of the type STRING or RAWSTRING cannot be used with aggregate functions.

Related Content

http://help.sap.com/saphelp_nw04/helpdata/en/fc/eb3990358411d1829f0000e829fbfe/content.htm

http://help.sap.com/abapdocu_70/en/ABAPSELECT_AGGREGATE.htm

http://abapreports.blogspot.com/2008/06/aggregate-functions-usage-in-abap.html

For more information, visit the ABAP 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.