Natural System Functions for Use in Processing Loops

This chapter describes those Natural system functions which can be used in a program loop context.

It covers the following topics:

- Using System Functions in Processing Loops

- AVER(r)(field)

- COUNT(r)(field)

- MAX(r)(field)

- MIN(r)(field)

- NAVER(r)(field)

- NCOUNT(r)(field)

- NMIN(r)(field)

- OLD(r)(field)

- SUM(r)(field)

- TOTAL(r)(field)

- Examples

Using System Functions in Processing Loops

The following topics are covered:

- Specification/Evaluation

- Use in SORT GIVE Statement

- Arithmetic Overflows in AVER, NAVER, SUM or TOTAL

- Statement Referencing (r)

Specification/Evaluation

Natural system functions may be specified in

- assignment and arithmetic statements:

- ○ MOVE

- ○ ASSIGN

- ○ COMPUTE

- ○ ADD

- ○ SUBTRACT

- ○ MULTIPLY

- ○ DIVIDE

- ● input/output statements:

  - ○ DISPLAY

  - ○ PRINT

  - ○ WRITE

that are used within any of the following statement blocks:

- ● AT BREAK

- ● AT END OF DATA

- ● AT END OF PAGE

that is, for all FIND, READ, HISTOGRAM, SORT or READ WORK FILE processing loops.

If a system function is used within an AT END OF PAGE statement, the corresponding DISPLAY statement must include the GIVE SYSTEM FUNCTIONS clause.

Records rejected by a WHERE clause are not evaluated by a system function.

If system functions are evaluated from database fields which originated from different levels of processing loops initiated with a FIND, READ, HISTOGRAM or SORT statement, the values are always processed according to their position in the loop hierarchy. For example, values for an outer loop will only be processed when new data values have been obtained for that loop.

If system functions are evaluated from user-defined variables, the processing is dependent on the position in the loop hierarchy where the user-defined variable was introduced in reporting mode. If the user-defined variable is defined before any processing loop is initiated, it will be evaluated for system functions in the loop where the AT BREAK, AT END OF DATA or AT END OF PAGE statement is defined. If a user-defined variable is introduced within a processing loop it will be processed the same as a database field from that processing.

For selective referencing of system function evaluation for user-defined variables it is recommended to specify a loop reference with the user-defined variable to indicate in which loop the value is to be processed. The loop reference may be specified as a statement label or source code line number.

Use in SORT GIVE Statement

System functions may also be referenced when they have been evaluated in a GIVE clause of a SORT statement.

For a reference to a system function evaluated with a SORT GIVE statement, the name of the system function must be prefixed with an asterisk (*).

Arithmetic Overflows in AVER, NAVER, SUM or TOTAL

Fields to which the system functions AVER, NAVER, SUM and TOTAL are to be applied must be long enough (either by default or user-specified) to hold any overflow digits. If any arithmetic overflow occurs, an error message will be issued.

Normally, the length is the same as that of the field to which the system function is applied; if this is not long enough, use the NL option of the SORT GIVE statement to increase the output length as follows:

```
SUM(field)(NL=nn)
```

This will not only increase the output length but also causes the field to be made longer internally.

Statement Referencing (r)

Statement referencing is also available for system functions (see also *Referencing of Database Fields Using (r) Notation* in the section *User-Defined Variables* of the *Programming Guide*).

By using a statement label or the source-code line number (*r*) you can determine in which processing loop the system function is to be evaluated for the specified field.

AVER(*r*)(*field*)

Format/length:	Same as field.
	Exception: for a field of format N, AVER*(field)* will be of format P (with the same length as the field).

This system function contains the average of all values encountered for the field specified with AVER. AVER is updated when the condition under which AVER was requested is true.

COUNT(*r*)(*field*)

Format/length:	P7

COUNT is incremented by 1 on each pass through the processing loop in which it is located. COUNT is incremented regardless of the value of the field specified with COUNT.

MAX(*r*)(*field*)

Format/length: Same as field.

This system function contains the maximum value encountered for the field specified with MAX. MAX is updated (if appropriate) each time the processing loop in which it is contained is executed.

MIN(*r*)(*field*)

Format/length: Same as field.

This system function contains the minimum value encountered for the field specified with MIN. MIN is updated (if appropriate) each time the processing loop in which it is located is executed.

NAVER(*r*)(*field*)

Format/length: Same as field.

> Exception: for a field of format N, NAVER*(field)* will be of format P (with the same length as the field).

This system function contains the average of all values - excluding null values - encountered for the field specified with NAVER. NAVER is updated when the condition under which NAVER was requested is true.

NCOUNT(*r*)(*field*)

Format/length: P7

NCOUNT is incremented by 1 on each pass through the processing loop in which it is located unless the value of the field specified with NCOUNT is a null value.

Whether the result of NCOUNT is an array or a scalar value depends on its argument (field). The number of the resulting occurrences is the same as of field.

NMIN(*r*)(*field*)

Format/length: Same as field.

This system function contains the minimum value encountered - excluding null values - for the field specified with NMIN. NMIN is updated (if appropriate) each time the processing loop in which it is located is executed.

OLD(*r*)(*field*)

Format/length: Same as field.

This system function contains the value which the field specified with `OLD` contained prior to a control break as specified in an `AT BREAK` condition, or prior to the end-of-page or end-of-data condition.

SUM(*r*)(*field*)

Format/length: Same as field.

Exception: for a field of format N, `SUM(field)` will be of format P (with the same length as the field).

This system function contains the sum of all values encountered for the field specified with `SUM`. `SUM` is updated each time the loop in which it is located is executed. When `SUM` is used following an `AT BREAK` condition, it is reset after each value break. Only values that occur between breaks are added.

TOTAL(*r*)(*field*)

Format/length: Same as field.

Exception: for a field of format N, `TOTAL(field)` will be of format P (with the same length as the field).

This system function contains the sum of all values encountered for the field specified with `TOTAL` in all open processing loops in which `TOTAL` is located.

Examples

The following examples are provided below:

- Example 1 - AT BREAK Statement with Natural System Functions OLD, MIN, AVER, MAX, SUM, COUNT

- Example 2 - AT BREAK Statement with Natural System Function AVER

- Example 3 - AT END OF DATA Statement with System Functions MAX, MIN, AVER

- Example 4 - AT END OF PAGE Statement with System Function AVER

Example 1 - AT BREAK Statement with Natural System Functions OLD, MIN, AVER, MAX, SUM, COUNT

```
** Example 'ATBEX3': AT BREAK  (with Natural system functions)
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
```

```
  2 CITY
  2 SALARY    (1)
  2 CURR-CODE (1)
END-DEFINE
*
LIMIT 3
READ EMPLOY-VIEW LOGICAL BY CITY = 'SALT LAKE CITY'
  DISPLAY NOTITLE CITY NAME 'SALARY' SALARY(1) 'CURRENCY' CURR-CODE(1)
  /*
  AT BREAK OF CITY
    WRITE /  OLD(CITY) (EM=X^X^X^X^X^X^X^X^X^X^X^X^X^X^X)
      31T '  MINIMUM:' MIN(SALARY(1))  CURR-CODE(1) /
      31T '  AVERAGE:' AVER(SALARY(1)) CURR-CODE(1) /
      31T '  MAXIMUM:' MAX(SALARY(1))  CURR-CODE(1) /
      31T '      SUM:' SUM(SALARY(1))  CURR-CODE(1) /
      35T COUNT(SALARY(1)) 'RECORDS FOUND' /
  END-BREAK
  /*
  AT END OF DATA
    WRITE 22T 'TOTAL (ALL RECORDS):'
            T*SALARY TOTAL(SALARY(1))  CURR-CODE(1)
  END-ENDDATA
END-READ
*
END
```

Output of program ATBEX3:

```
          CITY                  NAME            SALARY  CURRENCY
-------------------- -------------------- ---------- --------

SALT LAKE CITY       ANDERSON                   50000 USD
SALT LAKE CITY       SAMUELSON                  24000 USD

S A L T   L A K E   C I T Y      MINIMUM:      24000 USD
                                 AVERAGE:      37000 USD
                                 MAXIMUM:      50000 USD
                                     SUM:      74000 USD
                                    2 RECORDS FOUND

SAN DIEGO            GEE                        60000 USD

S A N   D I E G O                MINIMUM:      60000 USD
                                 AVERAGE:      60000 USD
                                 MAXIMUM:      60000 USD
                                     SUM:      60000 USD
                                    1 RECORDS FOUND

                     TOTAL (ALL RECORDS):    134000 USD
```

Example 2 - AT BREAK Statement with Natural System Function AVER

```
** Example 'ATBEX4': AT BREAK  (with Natural system functions)
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 CITY
  2 SALARY    (2)
*
1 #INC-SALARY (P11)
END-DEFINE
```

```
*
LIMIT 4
EMPL. READ EMPLOY-VIEW BY CITY STARTING FROM 'ALBU'
  COMPUTE #INC-SALARY = SALARY (1) + SALARY (2)
  DISPLAY NAME CITY SALARY (1:2) 'CUMULATIVE' #INC-SALARY
  SKIP 1
  /*
  AT BREAK CITY
    WRITE NOTITLE
      'AVERAGE:'            T*SALARY (1)  AVER(SALARY(1)) /
      'AVERAGE CUMULATIVE:' T*#INC-SALARY AVER(EMPL.) (#INC-SALARY)
  END-BREAK
END-READ
*
END
```

Output of program ATBEX4:

```
NAME                  CITY              ANNUAL    CUMULATIVE
                                                  SALARY
--------------------  ----------------  --------  ------------

HAMMOND               ALBUQUERQUE         22000        42200
                                          20200

ROLLING               ALBUQUERQUE         34000        65200
                                          31200

FREEMAN               ALBUQUERQUE         34000        65200
                                          31200

LINCOLN               ALBUQUERQUE         41000        78700
                                          37700

AVERAGE:                                  32750
AVERAGE CUMULATIVE:                                    62825
```

Example 3 - AT END OF DATA Statement with System Functions MAX, MIN, AVER

```
** Example 'AEDEX1S': AT END OF DATA
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 FIRST-NAME
  2 SALARY    (1)
  2 CURR-CODE (1)
END-DEFINE
*
LIMIT 5
EMP. FIND EMPLOY-VIEW WITH CITY = 'STUTTGART'
  IF NO RECORDS FOUND
    ENTER
  END-NOREC
  DISPLAY PERSONNEL-ID NAME FIRST-NAME
          SALARY (1) CURR-CODE (1)
  /*
  AT END OF DATA
    IF *COUNTER (EMP.) = 0
```

```
      WRITE 'NO RECORDS FOUND'
      ESCAPE BOTTOM
    END-IF
    WRITE NOTITLE / 'SALARY STATISTICS:'
                 / 7X 'MAXIMUM:' MAX(SALARY(1))  CURR-CODE (1)
                 / 7X 'MINIMUM:' MIN(SALARY(1))  CURR-CODE (1)
                 / 7X 'AVERAGE:' AVER(SALARY(1)) CURR-CODE (1)
  END-ENDDATA
  /*
END-FIND
*
END
```

Output of program `AEDEX1S`:

```
PERSONNEL           NAME               FIRST-NAME           ANNUAL   CURRENCY
   ID                                                       SALARY    CODE
--------- ------------------- ------------------- ---------- --------

11100328  BERGHAUS            ROSE                    70800 DM
11100329  BARTHEL             PETER                   42000 DM
11300313  AECKERLE            SUSANNE                 55200 DM
11300316  KANTE               GABRIELE                61200 DM
11500304  KLUGE               ELKE                    49200 DM

SALARY STATISTICS:
      MAXIMUM:      70800 DM
      MINIMUM:      42000 DM
      AVERAGE:      55680 DM
```

Example 4 - AT END OF PAGE Statement with System Function AVER

```
** Example 'AEPEX1S': AT END OF PAGE (structured mode)
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 JOB-TITLE
  2 SALARY    (1)
  2 CURR-CODE (1)
END-DEFINE
*
FORMAT PS=10
LIMIT 10
READ EMPLOY-VIEW BY PERSONNEL-ID FROM '20017000'
  DISPLAY NOTITLE GIVE SYSTEM FUNCTIONS
        NAME JOB-TITLE 'SALARY' SALARY(1) CURR-CODE (1)
  /*
  AT END OF PAGE
    WRITE / 28T 'AVERAGE SALARY: ...' AVER(SALARY(1)) CURR-CODE (1)
  END-ENDPAGE
END-READ
*
END
```

Output of program `AEPEX1S`:

```
      NAME                 CURRENT              SALARY   CURRENCY
                           POSITION                       CODE
--------------------  ------------------------  ---------- --------

CREMER                ANALYST                     34000 USD
MARKUSH               TRAINEE                     22000 USD
GEE                   MANAGER                     39500 USD
KUNEY                 DBA                         40200 USD
NEEDHAM               PROGRAMMER                  32500 USD
JACKSON               PROGRAMMER                  33000 USD

                AVERAGE SALARY:  ...             33533 USD
```