Systemfunktionen für Verarbeitungsschleifen

Dieses Kapitel erläutert die Natural-Systemfunktionen, die im Zusammenhang mit einer Programmschleife verwendet werden können.

Folgende Themen werden behandelt:

- Systemfunktionen für Verarbeitungsschleifen benutzen
- 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)
- Beispiele

Systemfunktionen für Verarbeitungsschleifen benutzen

Folgende Themen werden behandelt:

- Spezifikation/Auswertung
- Systemfunktionen im SORT GIVE-Statement
- Arithmetischer Überlauf bei AVER, NAVER, SUM oder TOTAL
- Statement-Referenzierung (r)

Spezifikation/Auswertung

Natural-Systemfunktionen können angegeben werden in

- zuweisenden und arithmetischen Statements:
  - `MOVE`

  - ASSIGN
  - COMPUTE
  - ADD
  - SUBTRACT
  - MULTIPLY
  - DIVIDE
- in Eingabe-Ausgabe-Statements:
  - DISPLAY
  - PRINT
  - WRITE

die in einem der folgenden Statement-Blöcke stehen:

- AT BREAK
- AT END OF DATA
- AT END OF PAGE

d.h. für alle Verarbeitungsschleifen in den Statements FIND, READ, HISTOGRAM, SORT oder READ WORK FILE.

Wenn eine Systemfunktion in einem AT END OF PAGE-Statement verwendet wird, muss das betreffende DISPLAY-Statement eine GIVE SYSTEM FUNCTIONS-Klausel enthalten.

Datensätze, die aufgrund einer WHERE-Klausel zurückgewiesen werden, werden von einer Systemfunktion nicht ausgewertet.

Wenn mittels Systemfunktionen Datenbankfelder ausgewertet werden, die aus FIND-, READ-, HISTOGRAM- oder SORT-Schleifen stammen, die auf verschiedenen Ebenen liegen, werden die Feldwerte jeweils entsprechend ihrer Position in der Verarbeitungsschleifen-Hierarchie verarbeitet. Werte einer äußeren Schleife werden zum Beispiel nur dann verarbeitet, wenn für diese Schleife neue Datenwerte erhalten werden.

Wird eine Benutzervariable vor der ersten Verarbeitungsschleife definiert, wird sie für Systemfunktionen in der Schleife ausgewertet, in der das AT BREAK-, AT END OF DATA- oder AT END OF PAGE-Statement steht; wird eine Benutzervariable innerhalb einer Schleife definiert, wird sie in der gleichen Weise wie ein Datenbankfeld in der derselben Schleife behandelt.

Bei dem selektiven Einsatz von Systemfunktionen zur Auswertung von Benutzervariablen empfiehlt es sich, eine bestimmte Verarbeitungsschleife zu referenzieren (mittels Statement-Label oder Sourcecode-Zeilennummer), um genau festzulegen, in welcher Schleife der Wert der Benutzervariablen ausgewertet werden soll.

Systemfunktionen im SORT GIVE-Statement

Eine Systemfunktion kann auch referenziert werden, nachdem sie in der GIVE-Klausel eines SORT-Statements ausgewertet wurde.

In diesem Fall muss dem Namen der Systemfunktion bei der Referenzierung ein Stern (*) vorangestellt werden.

Arithmetischer Überlauf bei AVER, NAVER, SUM oder TOTAL

Ein Feld, das Sie mit den Systemfunktionen AVER, NAVER, SUM oder TOTAL verwenden, muss lang genug sein (standardmäßig oder vom Benutzer definiert), um die Summe der Feldwerte aufzunehmen. Falls der addierte Wert die Länge des Feldes überschreitet, erhalten Sie eine Fehlermeldung.

Normalerweise hat die Systemfunktion dieselbe Länge wie das angegebene Feld; ist diese Länge nicht ausreichend, dann verwenden Sie den Parameter NL des SORT GIVE-Statements, um die Ausgabelänge zu vergrößern:

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

Dadurch vergrößert sich nicht nur die Ausgabelänge, sondern auch die interne Länge des Feldes.

Statement-Referenzierung (*r*)

Statement-Referenzierung kann auch auf bei Systemfunktionen angewendet werden. Weitere Einzelheiten entnehmen Sie dem Unterabschnitt *Datenbankfelder mit der (r)-Notation referenzieren - Notation (r)* im Abschnitt *Benutzervariablen* im *Natural Leitfaden zur Programmierung*.

Durch Verwendung eines Statement-Labels oder Angabe der Sourcecode-Zeilennummer (*r*) können Sie bestimmen, in welcher Verarbeitungsschleife die jeweilige Systemfunktion für das betreffende Feld ausgewertet werden soll.

AVER(*r*)(*field*)

Format/Länge: Wie Feld.

Ausnahme: bei einem Feld mit Format N hat AVER*(field)* Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält den Durchschnittswert (Average) aller Werte des angegebenen Feldes. AVER wird aktualisiert, wenn die Bedingung, unter der AVER angefordert wurde, erfüllt ist.

COUNT(*r*)(*field*)

Format/Länge: P7

Diese Systemfunktion zählt die Durchläufe einer Verarbeitungsschleife. Der Wert von COUNT erhöht sich jedesmal um 1, wenn die Verarbeitungsschleife, in der sich COUNT befindet, durchlaufen wird, und zwar unabhängig vom Wert des mit COUNT angegebenen Feldes.

MAX(*r*)(*field*)

Format/Länge: Wie Feld.

Diese Systemfunktion enthält den größten Wert des angegebenen Feldes. `MAX` wird (falls erforderlich) jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich `MAX` befindet, ausgeführt wird.

MIN(*r*)(*field*)

Format/Länge: Wie Feld.

Diese Systemfunktion enthält den kleinsten Wert des angegebenen Feldes. `MIN` wird (falls erforderlich) jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich `MIN` befindet, ausgeführt wird.

NAVER(*r*)(*field*)

Format/Länge: Wie Feld.

Ausnahme: bei einem Feld mit Format N hat `NAVER(field)` Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält den Durchschnittswert (Average) aller Werte des angegebenen Feldes, wobei Nullwerte nicht berücksichtigt werden. `NAVER` wird aktualisiert, wenn die Bedingung, unter der `NAVER` angefordert wurde, erfüllt ist.

NCOUNT(*r*)(*field*)

Format/Länge: P7

Diese Systemfunktion zählt die Durchläufe einer Verarbeitungsschleife. Der Wert von `NCOUNT` erhöht sich jedesmal um 1, wenn die Verarbeitungsschleife, in der sich `NCOUNT` befindet, durchlaufen wird, wobei Durchläufe, bei denen der Wert des angegebenen Feldes Null ist, nicht mitgezählt werden.

Ob das Ergebnis von `NCOUNT` ein Array oder ein Skalarwert ist, hängt vom Argument (*field*) ab. Die Anzahl der resultierenden Ausprägungen ist dieselbe wie bei Feld.

NMIN(*r*)(*field*)

Format/Länge: Wie Feld.

Diese Systemfunktion enthält den kleinsten Wert des angegebenen Feldes, wobei Nullwerte nicht berücksichtigt werden. `NMIN` wird (falls erforderlich) jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich `NMIN` befindet, ausgeführt wird.

OLD(*r*)(*field*)

Format/Länge: Wie Feld.

Diese Systemfunktion enthält den "alten" Wert des angegebenen Feldes, d.h. den Wert, den das Feld vor einem in einer AT BREAK-Bedingung spezifizierten Gruppenwechsel (Wechsel des Feldwertes) bzw. vor einer Seitenende- oder Datenende-Bedingung (END OF PAGE, END OF DATA) hatte.

SUM(*r*)(*field*)

Format/Länge: Wie Feld.

Ausnahme: bei einem Feld mit Format N hat SUM*(field)* Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält die Summe aller Werte des angegebenen Feldes. SUM wird jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich SUM befindet, ausgeführt wird. SUM wird nach jedem AT BREAK-Gruppenwechsel wieder auf Null gesetzt, addiert also nur Werte zwischen zwei Gruppenwechseln.

TOTAL(*r*)(*field*)

Format/Länge: Wie Feld.

Ausnahme: bei einem Feld mit Format N hat TOTAL*(field)* Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält die Gesamtsumme aller Werte des angegebenen Feldes in allen offenen Verarbeitungsschleifen, in denen TOTAL vorkommt.

Beispiele

- Beispiel 1 – AT BREAK-Statement mit Natural-Systemfunktionen OLD, MIN, AVER, MAX, SUM, COUNT
- Beispiel 2 – AT BREAK-Statement mit Natural-Systemfunktion AVER
- Beispiel 3 – AT END OF DATA-Statement mit Systemfunktionen MAX, MIN, AVER
- Beispiel 4 – AT END OF PAGE-Statement mit Systemfunktion AVER

Beispiel 1 – AT BREAK-Statement mit Natural-Systemfunktionen 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^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
```

Ausgabe des Programms 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
```

Beispiel 2 – AT BREAK-Statement mit Natural-Systemfunktion 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
```

Ausgabe des Programms 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
```

Beispiel 3 – AT END OF DATA-Statement mit Systemfunktionen 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
```

Ausgabe des Programms 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
```

Beispiel 4 – AT END OF PAGE-Statement mit Systemfunktion 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
```

Ausgabe des Programms 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
```