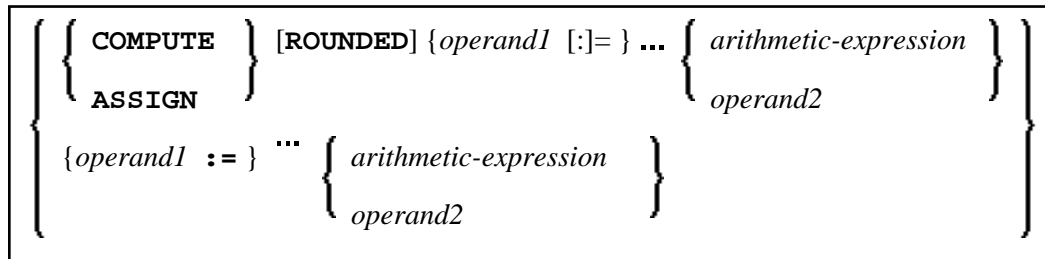
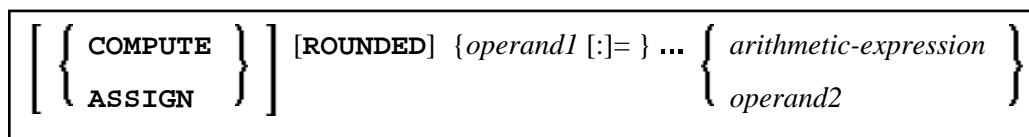


COMPUTE

Structured Mode-Syntax



Reporting Mode-Syntax



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Ergebnissenauigkeit einer Division
- SUBSTRING-Option
- Beispiele

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

Verwandte Statements: ADD | COMPRESS | DIVIDE | EXAMINE | MOVE | MOVE ALL | MULTIPLY | RESET | SEPARATE | SUBTRACT

Gehört zur Funktionsgruppe: *Arithmetische Funktionen und Datenzuweisungen*

Funktion

Das Statement COMPUTE dient zur Ausführung einer arithmetischen Operation sowie dazu, einem oder mehreren Feldern einen Wert zuzuweisen.

Ein COMPUTE-Statement mit mehreren Zieloperanden (*operand1*) ist identisch mit den entsprechenden einzelnen COMPUTE-Statements, wenn der Ausgangsoperand (*operand2*) kein arithmetischer Ausdruck ist.

#TARGET1 := #TARGET2 := #SOURCE

ist identisch mit

```
#TARGET1 := #SOURCE
#TARGET2 := #SOURCE
```

Beispiel:

```
DEFINE DATA LOCAL
1 #ARRAY(I4/1:3) INIT <3,0,9>
1 #INDEX(I4)
1 #RESULT(I4)
END-DEFINE
*
#INDEX := 1
*
#INDEX :=                /* #INDEX is 3
#RESULT :=                /* #RESULT is 9
#ARRAY(#INDEX)
*
#INDEX := 2
*
#INDEX :=                /* #INDEX is 0
#ARRAY(3) :=            /* returns run time error NAT1316
#ARRAY(#INDEX)
END
```

Wenn der Ausgangsoperand ein arithmetischer Ausdruck ist, wird der Ausdruck ausgewertet und das Ergebnis in einer temporären Variablen abgelegt. Danach wird diese temporäre Variable den Zieloperanden zugeordnet.

```
#TARGET1 := #TARGET2 := #SOURCE1 + 1
is identical to
#TEMP := #SOURCE1 + 1
#TARGET1 := #TEMP
#TARGET2 := #TEMP
```

Beispiel:

```
DEFINE DATA LOCAL
1 #ARRAY(I4/1:3) INIT <2, 0, 9>
1 #INDEX(I4)
1 #RESULT(I4)
END-DEFINE
*
#INDEX := 1
*
#INDEX :=                /* #INDEX is 3
#RESULT :=                /* #RESULT is 3
#ARRAY(#INDEX) + 1
*
#INDEX := 2
*
#INDEX :=                /* #INDEX is 0
#ARRAY(3) :=            /* returns run time error NAT1316
#ARRAY(#INDEX)
END
```

Weitere Informationen siehe *Regeln für arithmetische Operationen* im Leitfaden zur Programmierung und dort insbesondere die folgenden Abschnitte:

- *Arithmetische Operationen mit Arrays*
- *Datenübertragung* (Informationen zur Kompatibilität der Datenübertragung und zu Regeln für die Datenübertragung)

Syntax-Beschreibung

Operanden-Definitionstabelle:

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
<i>operand1</i>	S A M	A U N P I F B D T L C G O	ja	ja
<i>operand2</i>	C S A N E	A U N P I F B D T L C G O	ja	nein

Syntax-Element-Beschreibung:

COMPUTE ASSIGN [:=]	<p>Sie können das Statement in Kurzform angeben und den Statement-Namen COMPUTE (bzw. ASSIGN) weglassen.</p> <p>Wenn Sie im Structured Mode den Statement-Namen weglassen, müssen Sie vor das Gleichheitszeichen (=) einen Doppelpunkt (:) schreiben.</p> <p>Verwenden Sie die ROUNDED-Option, müssen Sie den Statement-Namen angeben.</p>
ROUNDED	<p>Wenn Sie das Schlüsselwort ROUNDED angeben, wird der Wert auf- bzw. abgerundet, bevor er <i>operand1</i> zugewiesen wird.</p> <p>Die für das Runden gültigen Regeln finden Sie im Abschnitt <i>Regeln für arithmetische Operationen, Abschneiden und Runden von Feldwerten</i> im <i>Leitfaden zur Programmierung</i>.</p>

<i>operand1</i>	<p>Ergebnisfeld:</p> <p><i>operand1</i> nimmt das Ergebnis der arithmetischen Operation bzw. Zuweisung auf.</p> <p>Zur Genauigkeit des Ergebnisses siehe Abschnitt <i>Genauigkeit von Ergebnissen bei arithmetischen Operationen</i> im <i>Leitfaden zur Programmierung</i>.</p> <p>Wenn <i>operand1</i> ein Datenbankfeld ist, ändert sich der Wert des Feldes auf der Datenbank dadurch nicht.</p> <p>Falls <i>operand1</i> eine dynamische Variable ist, wird er bis zur Länge von <i>operand2</i> oder bis zur Länge des Ergebnisses der arithmetische Operation einschließlich der nachfolgenden Leerzeichen aufgefüllt, und die Länge von <i>operand1</i> wird dann entsprechend angepasst.</p> <p>Die aktuelle Länge einer dynamischen Variable kann durch die Systemvariable *LENGTH bestimmt werden.</p> <p>Allgemeine Informationen zu dynamischen Variablen entnehmen Sie dem Abschnitt <i>Dynamische und große Variablen benutzen</i>.</p>
-----------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<i>arithmetic-expression</i>	<p>Ein arithmetischer Ausdruck (<i>arithmetic-expression</i>) besteht aus einer oder mehreren Konstanten, Datenbankfeldern bzw. Benutzervariablen.</p> <p>Mathematische Natural-Funktionen (siehe <i>Systemfunktionen</i>-Dokumentation) können ebenfalls als arithmetische Operanden verwendet werden.</p> <p>Die in einem arithmetischen Ausdruck verwendeten Operanden müssen eines der folgenden Formate haben: N, P, I, F, D oder T.</p> <p>Zum Format der Operanden siehe auch unter <i>Formatwahl im Hinblick auf die Verarbeitungszeit</i> im <i>Leitfaden zur Programmierung</i>.</p> <p>Die folgenden Operatoren können verwendet werden:</p> <table border="0"> <thead> <tr> <th>Operator</th> <th>Symbol</th> </tr> </thead> <tbody> <tr> <td>Klammern</td> <td>()</td> </tr> <tr> <td>Potenzierung</td> <td>**</td> </tr> <tr> <td>Multiplikation</td> <td>*</td> </tr> <tr> <td>Division</td> <td>/</td> </tr> <tr> <td>Addition</td> <td>+</td> </tr> <tr> <td>Subtraktion</td> <td>-</td> </tr> </tbody> </table> <p>Jedem Operatorzeichen sollte jeweils mindestens ein Leerzeichen vor- und nachgestellt werden, damit es nicht zu Konflikten mit Variablennamen, die eines dieser Zeichen enthalten, kommen kann.</p> <p>Bei der Verarbeitung arithmetischer Operationen gilt folgende Reihenfolge:</p> <ol style="list-style-type: none"> 1. Klammerrechnung 2. Potenzrechnung 3. Multiplikation/Division (von links nach rechts) 4. Addition/Subtraktion (von links nach rechts) 	Operator	Symbol	Klammern	()	Potenzierung	**	Multiplikation	*	Division	/	Addition	+	Subtraktion	-
Operator	Symbol														
Klammern	()														
Potenzierung	**														
Multiplikation	*														
Division	/														
Addition	+														
Subtraktion	-														
<i>operand2</i>	<p>Ausgangsfeld:</p> <p><i>operand2</i> ist das Ausgangsfeld.</p> <p>Wenn <i>operand1</i> das Format C hat, kann <i>operand2</i> auch als eine Attribut-Konstante angegeben werden (siehe <i>Benutzerkonstanten</i> im <i>Leitfaden zur Programmierung</i>).</p>														

Ergebnisgenauigkeit einer Division

Die Genauigkeit (Anzahl der Dezimalstellen) des Ergebnisses einer Division in einem COMPUTE-Statement bestimmt sich entweder aus der Genauigkeit des ersten Operanden (Dividenden) oder der des ersten Ergebnisfeldes, je nachdem welche größer ist.

Bei einer Division von Ganzzahlen gilt dagegen folgendes: Die Ergebnisgenauigkeit einer Division von zwei Ganzzahl-Konstanten bestimmt sich aus der Genauigkeit des ersten Ergebnisfeldes; ist jedoch eine der beiden Ganzzahlen eine Variable, dann ist auch das Ergebnis eine Ganzzahl (d.h. ohne Dezimalstellen, ganz gleich welche Genauigkeit das Ergebnisfeld hat).

SUBSTRING-Option

Wenn die Operanden alphanumerisches, Unicode- oder binäres Format haben, können Sie die SUBSTRING-Option verwenden (in der gleichen Weise wie beim MOVE-Statement beschrieben), um *operand1* einen Teil von *operand2* zuzuweisen.

Beispiele

- Beispiel 1 — ASSIGN-Statement
- Beispiel 2 — COMPUTE-Statement

Beispiel 1 — ASSIGN-Statement

```
** Example 'ASGEX1S': ASSIGN (structured mode)
*****
DEFINE DATA LOCAL
1 #A (N3)
1 #B (A6)
1 #C (N0.3)
1 #D (N0.5)
1 #E (N1.3)
1 #F (N5)
1 #G (A25)
1 #H (A3/1:3)
END-DEFINE
*
ASSIGN #A = 5
ASSIGN #B = 'ABC'
ASSIGN #C = .45
ASSIGN #D = #E = -0.12345
ASSIGN ROUNDED #F = 199.999
#G := 'HELLO'
#H (1) := 'UVW'
#H (3) := 'XYZ'
*
WRITE NOTITLE '=' #A
WRITE '=' #B
WRITE '=' #C
WRITE '=' #D / '=' #E
WRITE '=' #F
WRITE '=' #G
WRITE '=' #H (1:3)
*
END
```

Ausgabe des Programms ASGEX1S:

```
#A:      5
#B: ABC
#C:   .450
#D:  -.12345
#E: -0.123
#F:   200
#G: HELLO
#H: UVW   XYZ
```

Äquivalentes Reporting-Mode-Beispiel: ASGEX1R.

Beispiel 2 — COMPUTE-Statement

```
** Example 'CPTEX1': COMPUTE
*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 SALARY      (1:2)
*
1 #A           (P4)
1 #B           (N3.4)
1 #C           (N3.4)
1 #CUM-SALARY (P10)
1 #I           (P2)
END-DEFINE
*
COMPUTE #A = 3 * 2 + 4 / 2 - 1
WRITE NOTITLE 'COMPUTE #A = 3 * 2 + 4 / 2 - 1' 10X '=' #A
*
COMPUTE ROUNDED #B = 3 -4 / 2 * .89
WRITE 'COMPUTE ROUNDED #B = 3 -4 / 2 * .89' 5X '=' #B
*
COMPUTE #C = SQRT (#B)
WRITE 'COMPUTE #C = SQRT (#B)' 18X '=' #C
*
LIMIT 1
READ EMPLOY-VIEW BY PERSONNEL-ID STARTING FROM '20017000'
  WRITE / 'CURRENT SALARY:'   4X SALARY (1)
  / 'PREVIOUS SALARY:'      4X SALARY (2)
  FOR #I = 1 TO 2
    COMPUTE #CUM-SALARY = #CUM-SALARY + SALARY (#I)
  END-FOR
  WRITE 'CUMULATIVE SALARY:' #CUM-SALARY
END-READ
*
END
```

Ausgabe des Programms CPTEX1:

```
COMPUTE #A = 3 * 2 + 4 / 2 - 1      #A:      7
COMPUTE ROUNDED #B = 3 -4 / 2 * .89 #B:     1.2200
COMPUTE #C = SQRT (#B)             #C:     1.1045

CURRENT SALARY:                    34000
PREVIOUS SALARY:                   32300
CUMULATIVE SALARY:                 66300
```