Benutzung und Struktur des DEFINE DATA-Statements

Das erste Statement in einem im Structured Mode geschriebenen Natural-Programm muss immer ein `DEFINE DATA`-Statement sein. In diesem Statement definieren Sie alle Felder — Datenbankfelder wie Benutzervariablen — die in dem Programm verwendet werden sollen.

Dieses Kapitel behandelt folgende Themen:

- Felddefinitionen im DEFINE DATA-Statement
- Felder innerhalb eines DEFINE DATA-Statements definieren
- Felder in einer separaten Data Area definieren
- Struktur eines DEFINE DATA-Statements — Level-Nummern
- Speicherplatzausrichtung

Informationen zur strukturellen Einrückung eines Source-Programms siehe Natural-Systemkommando `STRUCT`.

Felddefinitionen im DEFINE DATA-Statement

Alle im Programm zu verwendenden Felder *müssen* im `DEFINE DATA`-Statement definiert werden.

Es gibt zwei Möglichkeiten, die Felder zu definieren:

- Die Felder können innerhalb des `DEFINE DATA`-Statements selbst definiert werden (siehe unten).
- Die Felder können außerhalb des Programms in einer Local Data Area oder einer Global Data Area definiert werden, wobei das `DEFINE DATA`-Statement diese Data Area referenziert (siehe unten).

Felder, die von mehreren Programmen/Unterprogrammen benutzt werden, sollten in einer programmexternen Data Area definiert werden.

Im Hinblick auf eine klare Anwendungsstruktur empfiehlt es sich in der Regel, Felder in programmexternen Data Areas zu definieren.

Data Areas werden mit dem Data-Area-Editor erstellt und gepflegt, der in der *Editors*-Dokumentation beschrieben ist.

Im ersten der folgenden Beispiele sind die Felder innerhalb des `DEFINE DATA`-Statements im Programm selbst definiert. Im zweiten Beispiel sind die gleichen Felder in einer Local Data Area (LDA) definiert, und das `DEFINE DATA`-Statement enthält lediglich eine Referenz auf diese Data Area.

Felder innerhalb eines DEFINE DATA-Statements definieren

Das folgende Beispiel veranschaulicht, wie Felder innerhalb des Programms im DEFINE DATA-Statements selbst definiert werden können:

```
DEFINE DATA LOCAL
1 VIEWEMP VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
  2 PERSONNEL-ID
1 #VARI-A (A20)
1 #VARI-B (N3.2)
1 #VARI-C (I4)
END-DEFINE
...
```

Felder in einer separaten Data Area definieren

Das folgende Beispiel veranschaulicht, wie Felder außerhalb eines Programms in einer Local Data Area (LDA) definiert werden können:

Programm:

```
DEFINE DATA LOCAL
  USING LDA39
END-DEFINE
...
```

Vom Programm referenzierte Local Data Area LDA39:

Diese LDA enthält eine View (Datensicht, siehe auch weiter unten) mit Namen VIEWEMP, in der die vom Programm verwendeten Felder definiert sind.

```
I T L Name                            F Leng Index/Init/EM/Name/Comment
- - - --------------------------------- - ---- --------------------------------
  V 1 VIEWEMP                                 EMPLOYEES
    2 NAME                            A   20
    2 FIRST-NAME                      A   20
    2 PERSONNEL-ID                    A    8
    1 #VARI-A                         A   20
    1 #VARI-B                         N  3.2
    1 #VARI-C                         I    4
```

Struktur eines DEFINE DATA-Statements — Level-Nummern

Folgende Themen werden behandelt:

- Struktur und Gruppierung der Definitionen

- Level-Nummern in View-Definitionen
- Level-Nummern in Feldgruppen
- Level-Nummern in Redefinitionen

Struktur und Gruppierung der Definitionen

Level-Nummern werden innerhalb eines `DEFINE DATA`-Statements verwendet, um die Struktur und Gruppierung der Definitionen zu zeigen. Dies ist relevant bei

- View-Definitionen
- Feldgruppen
- Redefinitionen

Level-Nummern sind 1- oder 2-stellige Zahlen von 01 bis 99 (die vorangestellte Null kann weggelassen werden).

Im Allgemeinen sind Variablen-Definitionen auf Level 1.

Die Level-Angaben in View-Definitionen, Redefinitionen und Gruppen müssen lückenlos sein. Es dürfen keine Level-Nummern ausgelassen werden.

Level-Nummern in View-Definitionen

Wenn Sie einen View definieren, geben Sie den View-Namen auf Level 1 an und die Felder, aus denen der View besteht, auf Level 2. Näheres zu View-Definitionen finden Sie im Abschnitt *Datenbankzugriffe*.

Beispiel für Level-Nummern in einer View-Definition:

```
DEFINE DATA LOCAL
1 VIEWEMP VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
  2 BIRTH
...
END-DEFINE
```

Level-Nummern in Feldgruppen

Mit der Definition von Gruppen ist es möglich, eine Reihe aufeinanderfolgender Felder auf einfache Weise zu referenzieren. Wenn Sie mehrere Felder unter einem gemeinsamen Gruppennamen definieren, können Sie diese Felder später im Programm referenzieren, indem Sie statt der Namen der einzelnen Felder lediglich den Gruppennamen angeben.

Der Gruppenname muss auf Level 1 definiert werden und die in der Gruppe enthaltenen Felder jeweils einen Level darunter.

Für Gruppennamen gelten die gleichen Namenskonventionen wie für Benutzervariablen. Siehe auch *Namenskonventionen für Benutzervariablen* in der Dokumentation *Natural benutzen*.

Beispiel für Level-Nummern in einer Gruppe:

```
DEFINE DATA LOCAL
1 #FIELDA (N2.2)
1 #FIELDB (I4)
1 #GROUPA
  2 #FIELDC (A20)
  2 #FIELDD (A10)
  2 #FIELDE (N3.2)
1 #FIELDF (A2)
...
END-DEFINE
```

In diesem Beispiel sind die Felder #FIELDC, #FIELDD und #FIELDE unter dem gemeinsamen Gruppennamen #GROUPA definiert. Die anderen drei Felder sind nicht Teil der Gruppe. Bitte beachten Sie, dass #GROUPA nur als Gruppenname dient und selbst kein Feld ist (und daher auch keine Format-/Längendefinition hat).

Level-Nummern in Redefinitionen

Wenn Sie ein Feld redefinieren, muss die REDEFINE-Option auf dem gleichen Level sein wie die Definition des Feldes, das redefiniert wird; und die Felder, die sich aus der Redefinition ergeben, müssen einen Level darunter sein. Näheres zu Redefinitionen finden Sie unter *Felder redefinieren*.

Beispiel für Level-Nummern in Redefinition:

```
DEFINE DATA LOCAL
1 VIEWEMP VIEW OF STAFFDDM
  2 BIRTH
  2 REDEFINE BIRTH
    3 #YEAR-OF-BIRTH  (N4)
    3 #MONTH-OF-BIRTH (N2)
    3 #DAY-OF-BIRTH   (N2)
1 #FIELDA (A20)
1 REDEFINE #FIELDA
  2 #SUBFIELD1 (N5)
  2 #SUBFIELD2 (A10)
  2 #SUBFIELD3 (N5)
...
END-DEFINE
```

In diesem Beispiel wird das Datenbankfeld BIRTH als drei Benutzervariablen redefiniert, und die Benutzervariable #FIELDA wird als drei andere Benutzervariablen redefiniert.

Speicherplatzausrichtung

Der Speicherbereich, in dem alle benutzerdefinierten Variablen gespeichert werden, beginnt immer an einer Doppelwortgrenze.

Wird ein DEFINE DATA-Statement benutzt, werden alle Datenblöcke (z.B. LOCAL-, GLOBAL-Blöcke) an einer Doppelwortgrenze ausgerichtet, und alle hierarchischen Strukturen (View -Definitionen und Gruppen) auf Level 1 werden auf Vollwortgrenze ausgerichtet. Redefinitionen, Skalar- und Array-Variablen werden selbst dann nicht ausgerichtet, wenn sie auf Level 1 definiert sind.

Die Ausrichtung an den Grenzen innerhalb des Datenbereichs ist Sache des Benutzers und richtet sich nach der Reihenfolge, in der die Variablen für Natural definiert sind.