Natural-Programmiermodi

Dieses Kapitel beschreibt die zwei von Natural unterstützten Programmiermodi.

Die folgenden Themen werden behandelt:

- Sinn und Zweck
- Programmiermodus festlegen/ändern
- Funktionale Unterschiede

Sinn und Zweck

Natural bietet zwei Formen der Programmierung:

- Reporting Mode
- Structured Mode

Anmerkung:
Grundsätzlich empfiehlt es sich, ausschließlich im Structured Mode zu arbeiten, um klar strukturierte Anwendungen zu erhalten.

Reporting Mode

Der Reporting Mode eignet sich nur für die Erstellung einfacher Reports und Programme, die keine komplexe Daten- und Programmstruktur erfordern. (Falls Sie sich entschließen sollten, ein Programm im Reporting Mode zu schreiben, sollten Sie bedenken, dass kleine Programme schnell umfangreicher und komplexer werden können.)

Bitte achten Sie darauf, dass manche Natural-Statements nur im Reporting Mode verfügbar sind, wohingegen andere eine spezifische Struktur aufweisen, wenn Sie im Reporting Mode benutzt werden. Eine Übersicht der Statements, die im Reporting Mode verwendet werden können, entnehmen Sie dem Abschnitt *Statements im Reporting Mode* in der *Statements*-Dokumentation.

Structured Mode

Der Structured Mode ist für komplexe Anwendungen gedacht, bei denen es auf eine klare und sinnvoll gegliederte Programmstruktur ankommt. Wesentliche Vorteile des Structured Mode sind:

- Die Programme müssen strukturierter geschrieben werden und sind daher leichter zu lesen und folglich auch leichter zu pflegen.
- Da alle in einem Programm verwendeten Felder an einer zentralen Stelle definiert werden müssen (und nicht, wie im Reporting Mode, über das ganze Programm verstreut sein dürfen), wird der Überblick über die verwendeten Daten erheblich erleichtert.

Darüber hinaus zwingt der Structured Mode zu einer genaueren Anwendungsplanung, bevor es an das eigentliche Programmieren geht. Viele Fehler und Unzulänglichkeiten bei der Programmierung werden dadurch von vorneherein vermieden.

Eine Übersicht der im Structured Mode zu benutzenden Statements entnehmen Sie dem Abschnitt *Statements nach Funktionsgruppen* in der *Statements*-Dokumentation.

Programmiermodus festlegen/ändern

Der Standardprogrammiermodus wird vom Natural-Administrator mit dem Profilparameter `SM` festgelegt.

Weitere Informationen zum Profil- und Session-Parameter `SM`, entnehmen Sie dem Abschnitt *SM - Programming in Structured Mode* in der *Parameter-Referenz*-Dokumentation.

Sie können den vorgegebenen Modus mit dem Systemkommando `GLOBALS` und dem Session-Parameter `SM` ändern:

Structured Mode:	`GLOBALS SM=ON`
Reporting Mode:	`GLOBALS SM=OFF`

Weitere Informationen, wie Sie den Programmiermodus ändern können, finden Sie in folgenden Dokumenten: *SM - Programming in Structured Mode* in der *Parameter-Referenz*-Dokumentation.

Funktionale Unterschiede

Die wichtigsten funktionalen Unterschiede zwischen Reporting Mode und Structured Mode lassen sich wie folgt zusammenfassen:

- Syntax zum Beenden von Schleifen und funktionalen Blöcken
- Verarbeitungsschleife im Reporting Mode beenden
- Verarbeitungsschleife im Structured Mode beenden
- Platzierung von Datenelementen in einem Programm
- Datenbank-Referenzierung

Anmerkung:
Ausführliche Informationen zu funktionalen Unterschieden zwischen den zwei Modi finden Sie in der *Statements*-Dokumentation. Sie enthält verschiedene Syntax-Diagramme und Syntax-Elementbeschreibungen für jedes modus-sensitive Statement. Eine Funktionsübersicht der im Reporting Mode zu benutzenden Statements finden Sie im Abschnitt *Statements im Reporting Mode* im der *Statements*-Dokumentation.

Syntax zum Beenden von Schleifen und funktionalen Blöcken

Reporting Mode:	(CLOSE) LOOP und DO ... DOEND-Statements werden hierzu verwendet. END-...-Statements (außer END-DEFINE, END-DECIDE und END-SUBROUTINE) können nicht benutzt werden.
Structured Mode:	Jede Schleife oder logische Verarbeitungsbedingung muss ausdrücklich mit einem entsprechenden END–...-Statement abgeschlossen werden. Dadurch wird sofort deutlich, wo welche Schleife bzw. Bedingung aufhört. LOOP und DO/DOEND-Statements können nicht benutzt werden.

Die beiden folgenden Beispiele veranschaulichen die je nach Programmiermodus unterschiedliche Konstruktion von Verarbeitungsschleifen und logischen Bedingungen.

Beispiel — Reporting Mode:

Im Reporting-Mode-Beispiel werden die Statements DO und DOEND verwendet, um den Statement-Block einzugrenzen, der an die AT END OF DATA-Bedingung geknüpft ist.

```
READ EMPLOYEES BY PERSONNEL-ID
DISPLAY NAME BIRTH
AT END OF DATA
   DO
      SKIP 2
      WRITE / 'LAST SELECTED:' OLD(NAME)
   DOEND
END
```

Das END-Statement beendet sämtliche aktiven Verarbeitungsschleifen.

Beispiel — Structured Mode:

Im Structured-Mode-Beispiel wird ein END-ENDDATA-Statement verwendet, um die AT END OF DATA-Bedingung zu beenden, sowie ein END-READ-Statement, um die READ-Schleife zu beenden. Das Ergebnis ist ein deutlicher strukturiertes Programm, in dem Sie sofort sehen können, wo welche Konstruktion anfängt und aufhört:

```
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 BIRTH

END-DEFINE
READ MYVIEW BY PERSONNEL-ID
   DISPLAY NAME BIRTH
   AT END OF DATA
      SKIP 2
      WRITE / 'LAST SELECTED:' OLD(NAME)
   END-ENDDATA
END-READ
END
```

Verarbeitungsschleife im Reporting Mode beenden

Zum Beenden einer Verarbeitungsschleife können Sie im Reporting Mode die Statements END, LOOP (bzw. CLOSE LOOP) oder SORT verwenden.

Mit dem LOOP-Statement können Sie mehrere Schleifen gleichzeitig schließen. Mit dem END-Statement können Sie sämtliche noch nicht beendeten Schleifen schließen. Diese Möglichkeit, mehrere Schleifen mit einem einzigen Statement zu beenden, stellt einen grundlegenden Unterschied zum Structured Mode dar.

Ein SORT-Statement beendet alle Schleifen und initiiert gleichzeitig eine neue Schleife.

Beispiel 1 — LOOP:

```
FIND ...
   FIND ...
   ...
   ...
   LOOP      /* closes inner FIND loop
LOOP         /* closes outer FIND loop
...
...
```

Beispiel 2 — END:

```
FIND ...
   FIND ...
   ...
   ...
END             /* closes all loops and ends processing
```

Beispiel 3 — SORT:

```
FIND ...
   FIND ...
   ...
   ...
SORT ...        /* closes all loops, initiates loop
...
END             /* closes SORT loop and ends processing
```

Verarbeitungsschleife im Structured Mode beenden

Im Structured Mode gibt es zum Beenden jeder Verarbeitungsschleife ein bestimmtes Statement. Mit dem END-Statement werden keine Schleifen geschlossen. Bei Verwendung des SORT-Statements müssen Sie vorher ein END-ALL-Statement verwenden, sowie zum Beenden der SORT-Schleife ein END-SORT-Statement.

Beispiel 1 — FIND:

```
FIND ...
   FIND ...
   ...
   ...
   END-FIND       /* closes inner FIND loop
END-FIND          /* closes outer FIND loop
...
```

Beispiel 2 — READ:

```
READ ...
   AT END OF DATA
   ...
   END-ENDDATA
   ...
END-READ           /* closes READ loop
...
...
END
```

Beispiel 3 — SORT:

```
READ ...
   FIND ...
   ...
   ...
END-ALL         /* closes all loops
SORT            /* opens loop
...
...
END-SORT        /* closes SORT loop
END
```

Platzierung von Datenelementen in einem Programm

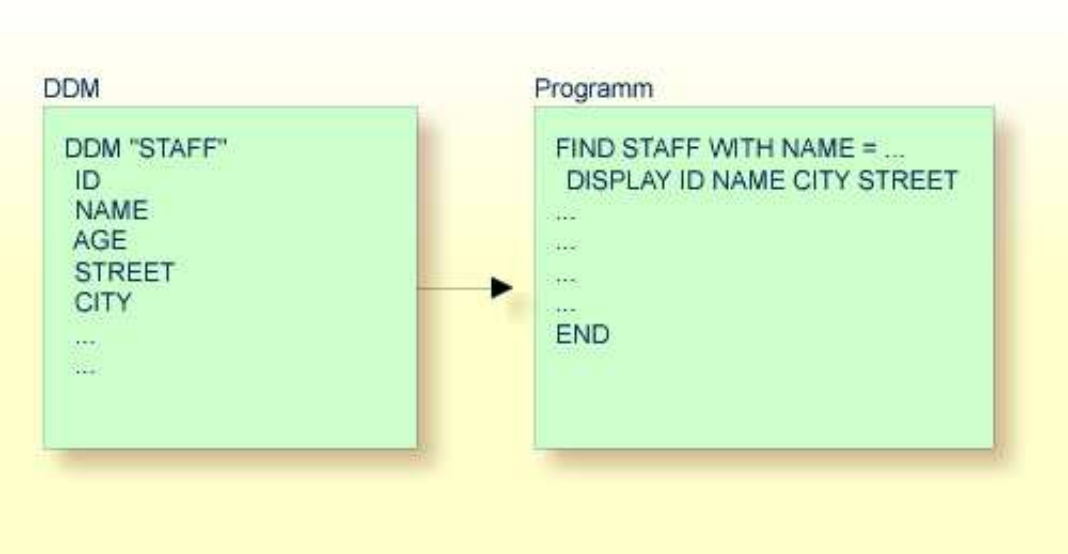
Im Reporting Mode können Datenbankfelder benutzt werden, ohne dass diese vorher in einem `DEFINE DATA`-Statement definiert werden müssen; außerdem können Benutzervariablen an jeder Stelle eines Programms, d.h. über das ganze Programm verstreut, definiert werden.

Im Structured Mode dagegen müssen *alle* verwendeten Datenelemente an einer zentralen Stelle (entweder im `DEFINE DATA`-Statement am Anfang des Programms oder in einer externen Data Area) definiert werden.

Datenbank-Referenzierung

Reporting Mode:

Im Reporting Mode ist es möglich, Datenbankfelder oder DDMs zu benutzen, ohne diese in einer Data Area definiert zu haben.



Structured Mode:

Im Structured Mode dagegen muss jedes Datenbankfeld, das benutzt werden soll, in einem `DEFINE DATA`-Statement angegeben werden (wie in den Abschnitten *Felder definieren* und *Datenbankzugriffe* beschrieben).