

FETCH

<pre> FETCH [{ REPEAT }] <i>operand1</i> [<i>operand2</i> [(<i>parameter</i>)]] ... RETURN </pre>

Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Beispiel

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

Verwandte Statements: CALL | CALL FILE | CALL LOOP | CALLNAT | DEFINE SUBROUTINE | ESCAPE | FETCH | PERFORM

Gehört zur Funktionsgruppe: *Aufrufen von Programmen und Unterprogrammen*

Funktion

Das Statement `FETCH` dient dazu, ein Natural-Objektprogramm auszuführen, welches als Hauptprogramm geschrieben wurde. Das zu ladende Programm muss vorher mit einem `STOW-` oder `CATALOG-`Kommando in der Natural-Systemdatei in Objektform gespeichert worden sein. Ein im Arbeitsbereich des Editors befindliches Sourceprogramm wird durch die Ausführung eines `FETCH`-Statements nicht überschrieben.

Für Natural RPC: Siehe *Notes on Natural Statements on the Server* in der *Natural Remote Procedure Call (RPC)*-Dokumentation.

Zusätzliche Anmerkungen

Zusätzlich zu den explizit mit dem `FETCH`-Statement übergebenen Parametern hat das aufgerufene Programm Zugang zu der Global Data Area des aufrufenden Programms.

Je nachdem, wie Ihr Natural-Administrator den Natural-Profilparameter `OPRB` (*Database Open/Close Processing*) gesetzt hat, kann es sein, dass das `FETCH`-Statement die Ausführung eines internen `END TRANSACTION`-Statements auslöst. Soll eine logische Transaktion mehrere Programme einschließen, so wenden Sie sich bitte vorher an Ihren Natural-Administrator, um sicherzustellen, dass der `OPRB`-Parameter entsprechend gesetzt ist.

Syntax-Beschreibung

Operanden-Definitionstabelle:

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

Syntax-Element-Beschreibung:

REPEAT	<p>Wegfall der Notwendigkeit einer Benutzer-Interaktion:</p> <p>REPEAT bewirkt, dass bei der Ausführung des aufgerufenen Programms der Benutzer bei INPUT-Statements keine Eingaben machen muss. REPEAT kann auch dazu eingesetzt werden, Informationen über die Ausführung des Programms am Bildschirm anzuzeigen, ohne dass der Benutzer daraufhin EINGABE drücken muss.</p>
RETURN	<p>Aufrufen und Ausführen eines Objekts des Typs Programm als Routine:</p> <p>Wird RETURN nicht angegeben, so wird das aufrufende Programm, welches das FETCH-Statement enthält, augenblicklich beendet, und das aufgerufene Programm wird als Hauptprogramm (Stufe 1) aktiviert.</p> <p>Geben Sie FETCH RETURN an, wird das aufrufende Programm nicht beendet, sondern nur unterbrochen, während das aufgerufene Programm als Unterprogramm auf einer höheren Stufe ausgeführt wird. Ein END- oder ESCAPE ROUTINE-Statement im aufgerufenen Programm bewirkt, dass die Kontrolle wieder an das aufrufende Programm übergeben wird, dessen Ausführung dann mit dem auf das FETCH RETURN folgende Statement fortgesetzt wird.</p>
<i>operand1</i>	<p>Programm-Name:</p> <p>Der Name des aufgerufenen Programms (maximal 8 Zeichen lang) kann entweder als alphanumerische Konstante oder als Inhalt einer alphanumerischen Variablen der Länge 1 bis 8 angegeben werden.</p> <p>Natural sucht das Programm zunächst in der zum Zeitpunkt der Ausführung des FETCH-Statements gerade aktiven Library. Wird es dort nicht gefunden, sucht Natural in den Steplibs. Wird das Programm auch dort nicht gefunden, gibt Natural eine entsprechende Fehlermeldung aus.</p> <p>Der Name des Programms darf ein Und-Zeichen (&) enthalten; zur Laufzeit wird dieses Zeichen durch den aus einem Zeichen bestehenden Code ersetzt, der dem aktuellen Wert der Systemvariablen *LANGUAGE entspricht. Dadurch ist es beispielsweise möglich, je nachdem in welcher Sprache eine Eingabe gemacht wird, zur Verarbeitung der Eingabe unterschiedliche Programme aufzurufen.</p>

operand2	<p>Zu übergebende Parameter-Felder:</p> <p>Mit dem FETCH-Statement können auch Parameterfelder an das aufgerufene Programm übergeben werden. Ein Parameterfeld kann mit einem beliebigen Format definiert werden; das Format wird dem jeweiligen INPUT-Feld entsprechend umgesetzt. Sämtliche Parameter werden oben auf dem Natural-Stack abgelegt.</p> <p>Das aufgerufene Programm liest die übergebenen Parameterfelder über ein INPUT-Statement. Das erste INPUT-Statement bewirkt, dass die Werte aller Parameterfelder in die mit dem INPUT-Statement angegebenen Felder übertragen werden. Da jedes mit einem numerischen Format definierte Parameterfeld eine Stelle für das Vorzeichen erhält, wenn sein Wert negativ ist, muss beim INPUT-Statement der Session-Parameter SG für die Parameterfelder auf SG=ON gesetzt werden.</p> <p>Werden mehr Parameter übergeben als vom INPUT-Statement gelesen werden können, so werden überschüssige Parameter ignoriert. Die Anzahl der Parameter kann mit Hilfe der Natural-Systemvariablen *DATA ermittelt werden.</p> <p>Anmerkung: Wenn <i>operand2</i> eine Zeitvariable (Format T) ist, wird nur die Zeitkomponente des Variableninhalts übergeben, aber nicht die Datumskomponente.</p>
parameter	<p>Datumsformat für Datumvariable:</p> <p>Wenn <i>operand2</i> eine Datumvariable ist, können Sie den Session-Parameter DF (Date Format) als <i>parameter</i> für diese Variable angeben.</p>

Beispiel

Aufrufendes Programm FETEX1:

```

** Example 'FETEX1': FETCH (with parameter)
*****
DEFINE DATA LOCAL
1 #PNUM (N8)
1 #FNC (A1)
END-DEFINE
*
INPUT 10X 'SELECTION MENU FOR EMPLOYEES SYSTEM' /
      10X '-' (35) //
      10X 'ADD (A)' /
      10X 'UPDATE (U)' /
      10X 'DELETE (D)' /
      10X 'STOP (.)' //
      10X 'PLEASE ENTER FUNCTION: ' #FNC ///
      10X 'PERSONNEL NUMBER:' #PNUM
*
DECIDE ON EVERY VALUE OF #FNC
  VALUE 'A', 'U', 'D'
  IF #PNUM = 0
    REINPUT 'PLEASE ENTER A VALID NUMBER' MARK *#PNUM
  END-IF
  VALUE 'A'
  FETCH 'FETEXAD' #PNUM
  VALUE 'U'
  FETCH 'FETEXUP' #PNUM

```

```

VALUE 'D'
  FETCH 'FETEXDE' #PNUM
VALUE '.'
  STOP
NONE
  REINPUT 'PLEASE ENTER A VALID FUNCTION' MARK *#FNC
END-DECIDE
*
END

```

Aufgerufenes Programm FETEXAD:

```

** Example 'FETEXAD': FETCH (called by FETEX1)
*****
DEFINE DATA LOCAL
1 #PERS-NR (N8)
END-DEFINE
*
INPUT #PERS-NR
*
WRITE *PROGRAM 'Record added with personnel number:' #PERS-NR
*
END

```

Aufgerufenes Programm FETEXUP:

```

** Example 'FETEXUP': FETCH (called by FETEX1)
*****
DEFINE DATA LOCAL
1 #PERS-NR (N8)
END-DEFINE
*
INPUT #PERS-NR
*
WRITE *PROGRAM 'Record updated with personnel number:' #PERS-NR
*
END

```

Aufgerufenes Programm FETEXDE:

```

** Example 'FETEXDE': FETCH (called by FETEX1)
*****
DEFINE DATA LOCAL
1 #PERS-NR (N8)
END-DEFINE
*
INPUT #PERS-NR
*
WRITE *PROGRAM 'Record deleted with personnel number:' #PERS-NR
*
END

```

Ausgabe des Programms FETEX1:

SELECTION MENU FOR EMPLOYEES SYSTEM

ADD (A)
UPDATE (U)
DELETE (D)
STOP (.)

PLEASE ENTER FUNCTION: D

PERSONNEL NUMBER: 1150304

Nach Eingabe und Bestätigung der Funktion und Personalnummer:

Page 1

05-01-13 11:58:46

FETEXDE Record deleted with personnel number: 1150304