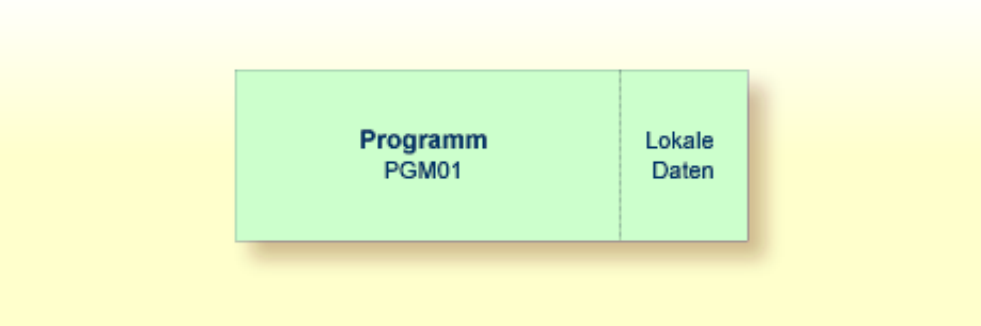
Datenbankzugriff

Sie werden nun ein kurzes Programm schreiben, mit dem bestimmte Daten aus einer Datenbankdatei gelesen und angezeigt werden.

Wenn Sie mit den Übungen in diesem Kapitel fertig sind, wird Ihre Beispielanwendung aus einem einzigen Modul bestehen (die Datenfelder, die vom Programm benutzt werden, sind direkt in diesem Programm definiert).



Dieses Kapitel enthält die folgenden Übungen:

- Programm unter einem neuen Namen speichern
- Benötigte Daten mit einem View definieren
- Daten aus einer Datenbank lesen
- Bestimmte Daten aus einer Datenbank lesen

Programm unter einem neuen Namen speichern

Sie werden jetzt ein neues Programm erstellen, das im weiteren Verlauf dieses Tutorials benutzt wird. Es wird erstellt, indem Sie Ihr "Hello World"-Programm unter einem neuen Namen speichern.

Programm unter einem neuen Namen speichern

1. Geben Sie eines der folgenden Kommandos in der Kommandozeile des Programmeditors ein:

```
SAVE PGM01
```

```
SA PGM01
```

   Das aktuelle Programm wird mit dem neuen Namen `PGM01` gespeichert. Das Programm mit dem Namen `HELLO` wird aber immer noch im Programmeditor angezeigt.

2. Geben Sie das folgende Kommando in der Kommandozeile des Programmeditors ein, um das neu erstellte Programm in den Programmeditor einzulesen:

```
READ PGM01
```

   Der im Programmeditor angezeigte Programmname ändert sich in PGM01.

3. Löschen Sie den gesamten Code im Programmeditor. Geben Sie hierzu das folgende Zeilenkommando am Anfang jeder zu löschenden Zeile ein und drücken Sie EINGABE:

```
.D
```

   Beispiel:

```
>                                       > +  Program     PGM01    Lib TUTORIAL
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 .DThe "Hello world!" example in Natural.
  0020 .D
  0030 .DSPLAY "Hello world!"
  0040 .DD /* End of program
  0050
```

   Oder:
   Geben Sie das folgende Zeilenkommando am Anfang der ersten Zeile ein und drücken Sie EINGABE:

```
.D(4)
```

   wobei die Zahl in Klammern für die Anzahl der zu löschenden Zeilen steht.

Benötigte Daten mit einem View definieren

Die Datenbankdatei und die Felder, die in Ihrem Programm benutzt werden sollen, müssen am Anfang des Programms zwischen DEFINE DATA und END-DEFINE angegeben werden.

Damit Natural auf den Inhalt einer Datenbankdatei zugreifen kann, wird eine logische Definition der physischen Datenbankdatei benötigt. Eine solche logische Dateidefinition wird "Data Definition Module" (DDM) genannt. Das DDM enthält Informationen über die einzelnen Felder der Datei. DDMs werden in der Regel vom Natural-Administrator definiert.

Damit die Datenbankfelder in einem Natural-Programm benutzt werden können, müssen Sie die Felder des DDMs in einem View angeben. In diesem Tutorial wird das DDM für die Datenbankdatei EMPLOYEES benutzt.

DEFINE DATA-Block angeben

- Geben Sie den folgenden Code im Programmeditor ein:

```
DEFINE DATA
LOCAL

END-DEFINE
*
END
```

LOCAL bedeutet, dass die Variablen, die Sie im nächsten Schritt definieren werden, lokale Variablen sind, die nur in diesem Programm zur Verfügung stehen.

Die Datenfelder des DDMs in einem geteilten Bildschirm (Split Screen) anzeigen

1. Geben Sie Folgendes in der Kommandozeile des Programmeditors ein:

```
SPLIT VIEW EMPLOYEES SHORT
```

SHORT bedeutet, dass die Datenfelder in der Kurzform aufgelistet werden sollen (das heißt: es werden nur die Adabas-Kurznamen und die entsprechenden Natural-Felder angezeigt).

Der Bildschirm ist jetzt in zwei Teile unterteilt. Die Datenfelder des DDMs werden im unteren Teil des Bildschirms angezeigt. Die Daten im unteren Teil des Bildschirms können nicht editiert werden.

```
>                                        > +  Program     PGM01    Lib TUTORIAL
All    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  0010 DEFINE DATA
  0020 LOCAL
  0040 END-DEFINE
  0050 *
  0060 END
  0070
  0080
  0090
  0100
  0110
       ....+....1....+....2....+....3....+....4....+....5....+... S 5    L 1
Split Top  View EMPLOYEES                    DBID     0 FNR     1 Def seq
    1  AA  PERSONNEL-ID                     A    8    D CNNNNNNN
  G 1  AB  FULL-NAME                                    NAME INFORMATION
    2  AC  FIRST-NAME                       A   20  N   FIRST/CHRISTIAN NA
    2  AD  MIDDLE-I                         A    1  N   MIDDLE INITIAL
    2  AE  NAME                             A   20    D SURNAME/FAMILY NAM
    1  AD  MIDDLE-NAME                      A   20  N   SECOND/MIDDLE NAME
    1  AF  MAR-STAT                         A    1  F   M=MARRIED
    1  AG  SEX                              A    1  F
    1  AH  BIRTH                            D    6  N D BIRTH-DATE (YYYY-M
```

2. Mit den folgenden Kommandos können Sie den View durchblättern, um zu überprüfen, welche Felder benutzt werden und wie sie definiert sind:

Kommando	Beschreibung
`SPLIT +` oder `S +`	Im View vorwärts blättern.
`SPLIT -` oder `S -`	Im View rückwärts blättern.
`SPLIT .` oder `S .`	Den Split-Screen-Modus beenden.

Im nächsten Schritt wird davon ausgegangen, dass der Split-Screen-Modus beendet wurde.

3. Stellen Sie den Cursor an die erste Position der Zeile, in der `LOCAL` steht und geben Sie Folgendes ein:

```
.I
```

   Im Vollbildmodus werden 9 Leerzeilen eingefügt. Im Split-Screen-Modus wären nur 4 Leerzeilen eingefügt worden.

4. Geben Sie unter `LOCAL` den folgenden Code ein:

```
1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
  2 FULL-NAME
    3 NAME (A20)
  2 DEPT (A6)
  2 LEAVE-DATA
    3 LEAVE-DUE (N2)
```

5. Drücken Sie EINGABE.

   Die übrigen Leerzeilen werden entfernt.

   Anmerkung:
   Die übrigen Leerzeilen werden nicht entfernt, wenn die Standardeinstellung im Editorprofil geändert wurde, das heißt: wenn die Option **Empty Line Suppression** auf "N" gesetzt wurde.

Die erste Zeile enthält den Namen des Views und den Namen der Datenbankdatei, aus der die Felder genommen werden. Dies wird immer auf der ersten Ebene (Level 1) definiert. Der Level wird am Anfang der Zeile definiert. Die Namen der Datenbankfelder aus dem DDM sind auf den Levels 2 und 3 definiert.

Levels werden im Zusammenhang mit Feldgruppierungen benutzt. Felder mit einem Level von 2 oder höher werden als Teil der direkt davor stehenden Gruppe angesehen, die auf einem niedrigeren Level steht. Die Definition einer Gruppe ermöglicht das Referenzieren von mehreren Feldern (dies kann auch nur ein einziges Feld sein), indem man den Gruppennamen benutzt. Dies ist eine komfortable und effiziente Methode zum Referenzieren von mehreren aufeinander folgenden Feldern.

Format und Länge jedes Feldes sind in Klammern angegeben. "A" steht für alphanumerisch und "N" steht für numerisch.

Daten aus einer Datenbank lesen

Jetzt, nachdem Sie alle erforderlichen Daten definiert haben, werden Sie eine READ-Schleife in Ihr Programm einfügen. Hiermit werden die Daten mit Hilfe des definierten Views aus der Datenbankdatei gelesen. Mit jeder Schleife wird ein Mitarbeiter aus der Datenbankdatei gelesen. Name, Abteilung (Department) und die restlichen Urlaubstage (Leave Due) für diesen Mitarbeiter werden angezeigt. Die Daten werden so lange gelesen, bis alle Mitarbeiter angezeigt wurden.

Anmerkung:
Es kann passieren, dass eine Fehlermeldung erscheint, die besagt, dass die letzte Transaktion aus der Datenbank zurückgezogen wurde (transaction backed out). Dies passiert in der Regel dann, wenn das Adabas-Zeitlimit für Nichtaktivität überschritten wurde. Wenn ein solcher Fehler auftritt, sollten Sie Ihre letzte Aktion einfach wiederholen (geben Sie zum Beispiel das Kommando RUN noch einmal ein).

Daten aus einer Datenbank lesen

1. Geben Sie folgendes unter END-DEFINE ein (benutzen Sie zum Einfügen von Leerzeilen das Kommando .I wie oben beschrieben):

   ```
   READ EMPLOYEES-VIEW BY NAME
   *
     DISPLAY NAME 3X DEPT 3X LEAVE-DUE
   *
   END-READ
   ```

   BY NAME bedeutet, dass die aus der Datenbank gelesenen Daten alphabetisch nach den Namen sortiert werden sollen.

   Mit dem DISPLAY-Statement wird die Ausgabe im Spaltenformat angeordnet. Für jedes angegebene Feld wird eine Spalte erzeugt und jede Spalte enthält eine Überschrift. 3X bedeutet, dass 3 Leerzeichen zwischen den Spalten eingefügt werden sollen.

2. Führen Sie das Programm mit RUN aus.

   Die folgende Ausgabe wird angezeigt.

   ```
   MORE
   Page      1                                                  05-05-18  16:06:49

           NAME           DEPARTMENT   LEAVE
                             CODE       DUE
   --------------------   ----------   -----

   ABELLAN                PROD04        20
   ACHIESON               COMP02        25
   ADAM                   VENT59        19
   ADKINSON               TECH10        38
   ADKINSON               TECH10        18
   ADKINSON               TECH05        17
   ADKINSON               MGMT10        28
   ADKINSON               TECH10        26
   ADKINSON               SALE30        36
   ADKINSON               SALE20        37
   ADKINSON               SALE20        30
   AECKERLE               SALE47        31
   AFANASSIEV             MGMT30        26
   ```

```
AFANASSIEV              TECH10        35
AHL                     MARK09        30
AKROYD                  COMP03        20
ALEMAN                  FINA03        20
```

Das DISPLAY-Statement sorgt dafür, dass die Spaltenüberschriften (die aus dem DDM genommen werden) unterstrichen sind und dass sich zwischen der Unterstreichung und den Daten eine Leerzeile befindet. Jede Spalte hat die Breite, die im DEFINE DATA-Block definiert wurde (das heißt: die Breite, die im View definiert ist).

Der Titel oben auf jeder Seite (mit Seitennummer, Datum und Uhrzeit) wird ebenfalls vom DISPLAY-Statement erzeugt.

3. Drücken Sie wiederholt EINGABE, um alle Seiten nacheinander anzuzeigen.

   Sie kehren zum Programmeditor zurück, nachdem alle Mitarbeiter angezeigt wurden.

   Tipp:
   Wenn Sie zum Programmeditor zurückkehren möchten, bevor alle Mitarbeiter angezeigt wurden, geben Sie EDIT oder dessen Abkürzung E in der MORE-Zeile ein. Sie können auch das Terminalkommando %. in der MORE-Zeile eingeben; dies bricht die aktuelle Natural-Verarbeitung ab. Standardmäßig beginnt jedes Terminalkommando mit einem Prozentzeichen (%) als Steuerzeichen. Es ist jedoch möglich, dass Ihr Administrator ein anderes Steuerzeichen definiert hat.

Bestimmte Daten aus einer Datenbank lesen

Da die Ausgabe im Moment sehr lang ist, werden Sie sie nun eingrenzen. Es sollen nur noch die Daten für den Namenbereich angezeigt werden, der mit "Adkinson" beginnt und mit "Bennett" endet. Diese Namen sind in der Demodatenbank definiert.

Ausgabe auf einen Namensbereich eingrenzen

1. Bevor Sie neue Variablen benutzen können, müssen Sie sie definieren. Geben Sie deshalb Folgendes unter LOCAL ein:

   ```
   1 #NAME-START        (A20) INIT <"ADKINSON">
   1 #NAME-END          (A20) INIT <"BENNETT">
   ```

   Dies sind Benutzervariablen; sie sind nicht in der Demodatenbank definiert. Das Rautenzeichen (#) am Anfang des Namens wird dazu benutzt, die Benutzervariablen von den Feldern in der Demodatenbank zu unterscheiden; es ist jedoch nicht zwingend notwendig, dieses Zeichen zu verwenden.

   INIT definiert den Vorgabewert für ein Feld. Der Vorgabewert muss in spitzen Klammern und Anführungszeichen angegeben werden.

2. Geben Sie Folgendes unter dem READ-Statement ein:

   ```
   STARTING FROM #NAME-START
   ENDING AT #NAME-END
   ```

   Ihr Programm sollte nun folgendermaßen aussehen:

```
DEFINE DATA
LOCAL
  1 #NAME-START        (A20) INIT <"ADKINSON">
  1 #NAME-END          (A20) INIT <"BENNETT">
  1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    2 FULL-NAME
      3 NAME (A20)
    2 DEPT (A6)
    2 LEAVE-DATA
      3 LEAVE-DUE (N2)
END-DEFINE
*
READ EMPLOYEES-VIEW BY NAME
  STARTING FROM #NAME-START
  ENDING AT #NAME-END
*
  DISPLAY NAME 3X DEPT 3X LEAVE-DUE
*
END-READ
*
END
```

Ihr Programmcode ist jetzt länger als eine Bildschirmseite. Um im Programm zu navigieren, können Sie die folgenden Kommandos oder Tasten benutzen:

Kommando	**Beschreibung**
BOT	Geht an das Ende des Programms.
TOP	Geht an den Anfang des Programms.
Taste	**Beschreibung**
PF8 oder EINGABE	Blättert eine Seite im Programm nach unten.
PF7	Blättert eine Seite im Programm nach oben.

3. Führen Sie das Programm mit RUN aus.

   Die Ausgabe wird angezeigt. Wenn Sie wiederholt EINGABE drücken, werden Sie bemerken, dass Sie nach wenigen Seiten zum Programmeditor zurückkehren (d.h. nachdem die Daten für den letzten Mitarbeiter mit dem Namen Bennett angezeigt wurden).

4. Speichern Sie das Programm mit STOW.

Sie können nun mit den nächsten Übungen fortfahren: *Benutzereingaben*.