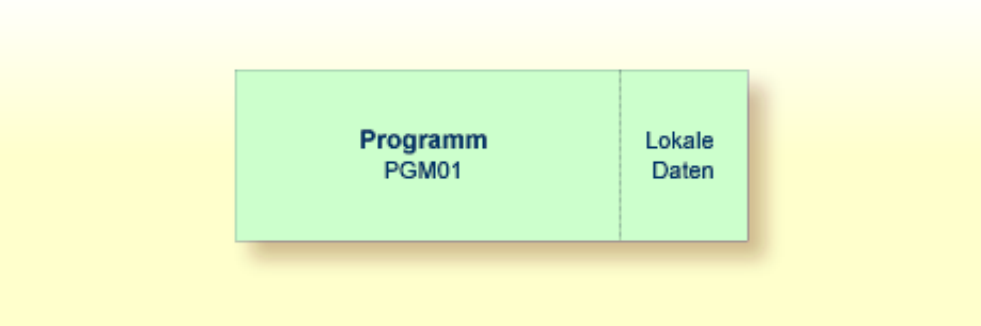
Datenbankzugriff

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:

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

Demodatenbank starten

Die Demodatenbank `SAG-DEMO-DB` wird nicht automatisch gestartet. Bevor Sie mit den Übungen in diesem Kapitel fortfahren können, müssen Sie sich davon überzeugen, dass die Datenbank gestartet wurde. Andernfalls funktionieren die Beispiele nicht.

Die folgende Beschreibung gilt wenn Adabas lokal unter Windows installiert wurde. Wenn Sie mit der UNIX-Version arbeiten möchten und die Datenbank nicht aktiv ist, müssen Sie Ihren Administrator bitten, sie zu starten.

Demodatenbank starten

1. Wählen Sie aus dem Menü **Start** den Befehl **Programme** > **Adabas *n.n*** > **DBA Workbench**.

   Der Status der Demodatenbank wird in der daraufhin erscheinenden Datenbankliste angezeigt. Wenn der Status "Active" angezeigt wird, sind keine weiteren Schritte erforderlich und Sie können das Anwendungsfenster der DBA Workbench schließen.

Wenn der Status "Active" nicht angezeigt wird, müssen Sie wie unten beschrieben fortfahren.

2. Markieren Sie **SAG-DEMO-DB** in der Datenbankliste.
3. Wählen Sie aus dem Menü **Database** den Befehl **Start**.

   Ein Dialogfeld erscheint mit dem Hinweis, dass die Datenbank gestartet wurde.
4. Wählen Sie die Befehlsschaltfläche **OK**, um das Dialogfeld zu schließen.
5. Schließen Sie das Anwendungsfenster der DBA Workbench.

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. Wählen Sie aus dem Menü **Object** den Befehl **Save As**.

   Tipp:
   Achten Sie darauf, dass der Programmeditor aktiv ist. Andernfalls steht der oben genannte Befehl nicht zur Verfügung.

   Das Dialogfeld **Save As** erscheint.
2. Geben Sie "PGM01" als neuen Namen für das Programm ein.
3. Wählen Sie die Befehlsschaltfläche **OK**.

   Der neue Name wird jetzt in der Titelleiste des Programmeditors angezeigt.

   Im Library-Workspace wird das neue Programm im Knoten **Programs** angezeigt. Da das Programm noch nicht mit `STOW` gespeichert wurde, enthält das Programmsymbol keinen grünen Ball.

4. Löschen Sie den gesamten Code im Programmeditor (zum Beispiel indem Sie den gesamten Text mit STRG+A markieren und dann mit der ENTF-Taste löschen - dies ist Windows-Standardfunktionalität).

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. Beispiel-DDMs stehen in der System-Library SYSEXDDM zur Verfügung. In diesem Tutorial wird das DDM für die Datenbankdatei EMPLOYEES benutzt.

Sie können die Felder, einschließlich Format- und Längenangaben, aus dem DDM in den Programmeditor importieren.

DEFINE DATA-Block angeben

- Geben Sie den folgenden Code im Programmeditor ein:

```
DEFINE DATA
LOCAL

END-DEFINE
*
END
```

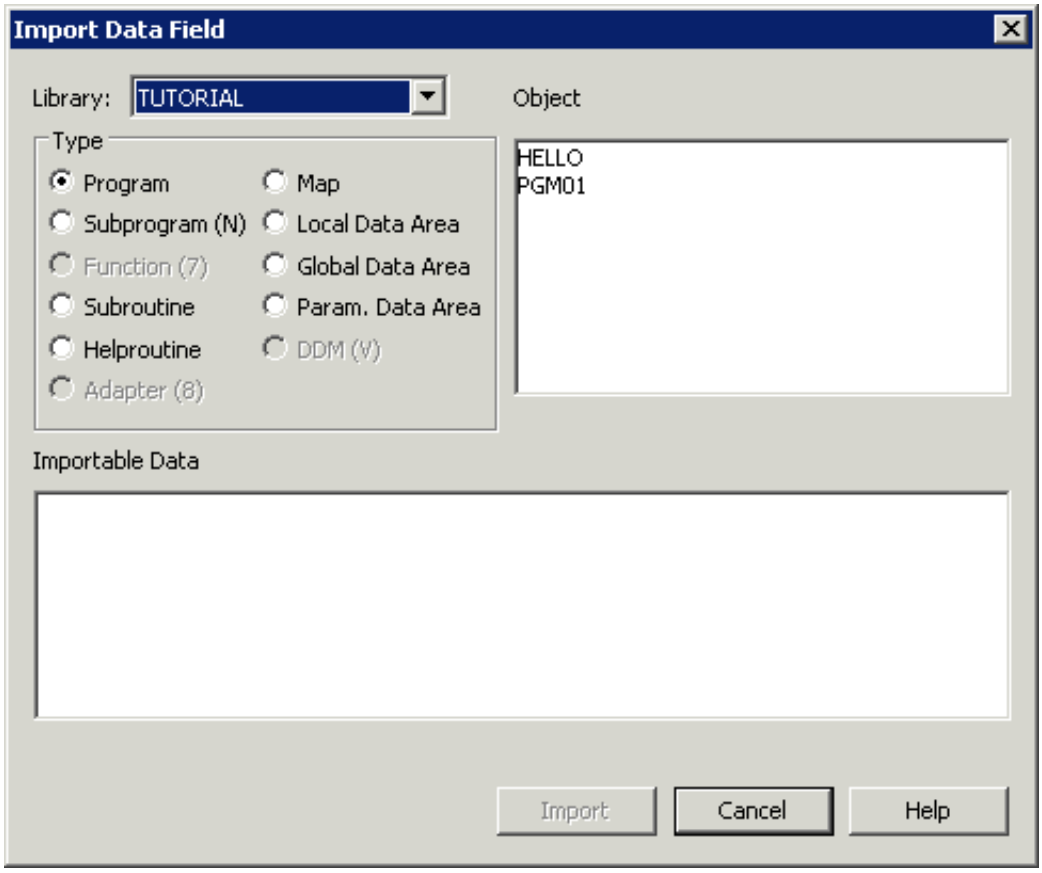
Tipp:
Die Windows-Version von Natural unterscheidet nicht zwischen Groß- und Kleinschreibung. Wenn Sie jedoch mit der Großrechner-Version von Natural arbeiten, dann werden Schlüsselwörter (Keywords) und Bezeichnungen (Identifiers) immer in Großbuchstaben angegeben; Textkonstanten können Kleinbuchstaben enthalten. Wenn Sie Ihr Programm also auch auf dem Großrechner editieren möchten, wird empfohlen, dass Sie den Programmcode so eingeben, wie Sie es auch auf einem Großrechner tun würden.

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

Datenfelder aus einem DDM importieren

1. Stellen Sie den Cursor in die Zeile unter LOCAL.
2. Wählen Sie aus dem Menü **Program** den Befehl **Import**.

   Das Dialogfeld **Import Data Field** erscheint.



3. Markieren Sie im Dropdown-Listenfeld **Library** den Eintrag **SYSEXDDM**.

   Wenn das Optionsfeld **DDM** markiert ist, werden alle definierten DDMs im Listenfeld **Object** angezeigt.

4. Markieren Sie das Beispiel-DDM mit dem Namen `EMPLOYEES`.

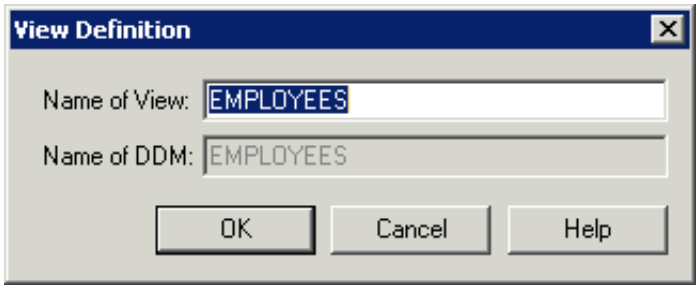
   Die importierbaren Daten werden jetzt unten im Dialogfeld angezeigt.

5. Drücken Sie STRG und markieren Sie die folgenden Felder:

   ```
   FULL-NAME
   NAME
   DEPT
   LEAVE-DATA
   LEAVE-DUE
   ```

6. Wählen Sie die Befehlsschaltfläche **Import**.

   Das Dialogfeld **View Definition** erscheint.



Standardmäßig wird der Name des DDMs als View-Name vorgeschlagen. Sie können aber auch einen beliebigen anderen Namen angeben.

7. Geben Sie "EMPLOYEES-VIEW" als View-Name ein.
8. Wählen Sie die Befehlsschaltfläche **OK**.

   Die Befehlsschaltfläche **Cancel** im Dialogfeld **Import Data Field** trägt nun den Namen **Quit**.
9. Wählen Sie die Befehlsschaltfläche **Quit**, um das Dialogfeld **Import Data Field** zu schließen.

   Der folgende Code wurde im Programmeditor eingefügt:

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

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 Transaktion abgebrochen wurde (transaction aborted). 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:

```
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.

```
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, drücken Sie ESC.

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
```

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*.