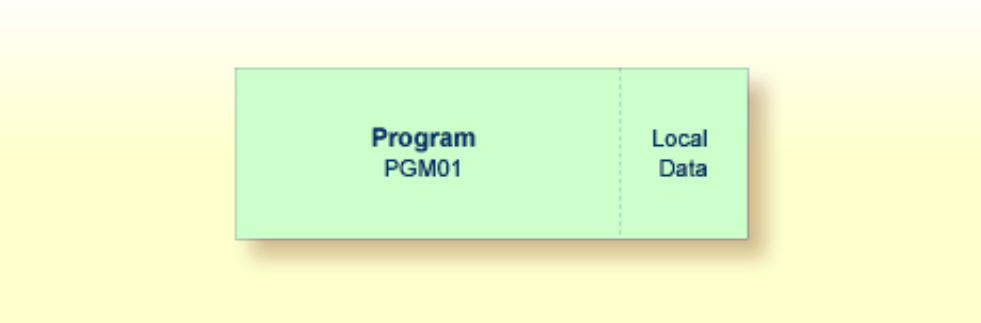
Database Access

You will now write a short program which reads specific data from a database file and displays the corresponding output.

When you have completed the exercises below, your sample application will consist of just one module (the data fields that are used by the program are defined within the program):



This chapter contains the following exercises:

- Starting the Demo Database
- Saving Your Program Under a New Name
- Defining the Required Data Using a View
- Reading Data from a Database
- Reading Selected Data from a Database

Starting the Demo Database

The demo database `SAG-DEMO-DB` is not started automatically. Before you can proceed with the exercises in this tutorial, you must make sure that it has been started. Otherwise, the examples will not work.

The following description applies when Adabas has been installed locally under Windows. If you want to work with the UNIX version and the demo database is not already running, ask your UNIX administrator to start it.

To start the demo database

1. From the **Start** menu, choose **Programs** > **Adabas *n.n*** > **DBA Workbench**.

   The status of the demo database is shown in the resulting database list. When the status is "Active", no further steps are required and you can close the DBA Workbench application window.

   When the status is not "Active", proceed as described below.

2. Select **SAG-DEMO-DB** in the database list.
3. From the **Database** menu, choose **Start**.

   A dialog box appears indicating that the database has been started.
4. Choose the **OK** button to close the dialog box.
5. Close the DBA Workbench application window.

Saving Your Program Under a New Name

You will now create a new program which will be used in the remainder of this tutorial. It will be created by saving your Hello World program under a new name.

To save the program under a new name

1. From the **Object** menu, choose **Save As**.

   Tip:
   Make sure that the program editor is selected. Otherwise the above command is not available.

   The **Save As** dialog box appears.
2. Specify "PGM01" as the new name for the program.
3. Choose the **OK** button.

   The new name is now shown in the title bar of the program editor.

   In the library workspace, the new program is shown in the **Programs** node. Since it has not yet been stowed, its program icon does not contain a green dot.

4. Delete all code in the program editor (for example, by pressing CTRL+A to select all text and then pressing the DEL key - this is standard Windows functionality).

Defining the Required Data Using a View

The database file and the fields that are to be used by your program have to be specified between `DEFINE DATA` and `END-DEFINE` at the top of the program.

For Natural to be able to access a database file, a logical definition of the physical database file is required. Such a logical file definition is called a data definition module (DDM). The DDM contains information about the individual fields of the file. DDMs are usually defined by the Natural administrator.

To be able to use the database fields in a Natural program, you must specify the fields from the DDM in a view. Sample DDMs are provided in the system library SYSEXDDM. For this tutorial, we will use the DDM for the EMPLOYEES database file.

You can import the fields, including the required format and length definitions, from the DDM into the program editor.

To specify the DEFINE DATA block

- Enter the following code in the program editor:

```
DEFINE DATA
LOCAL

END-DEFINE
*
END
```

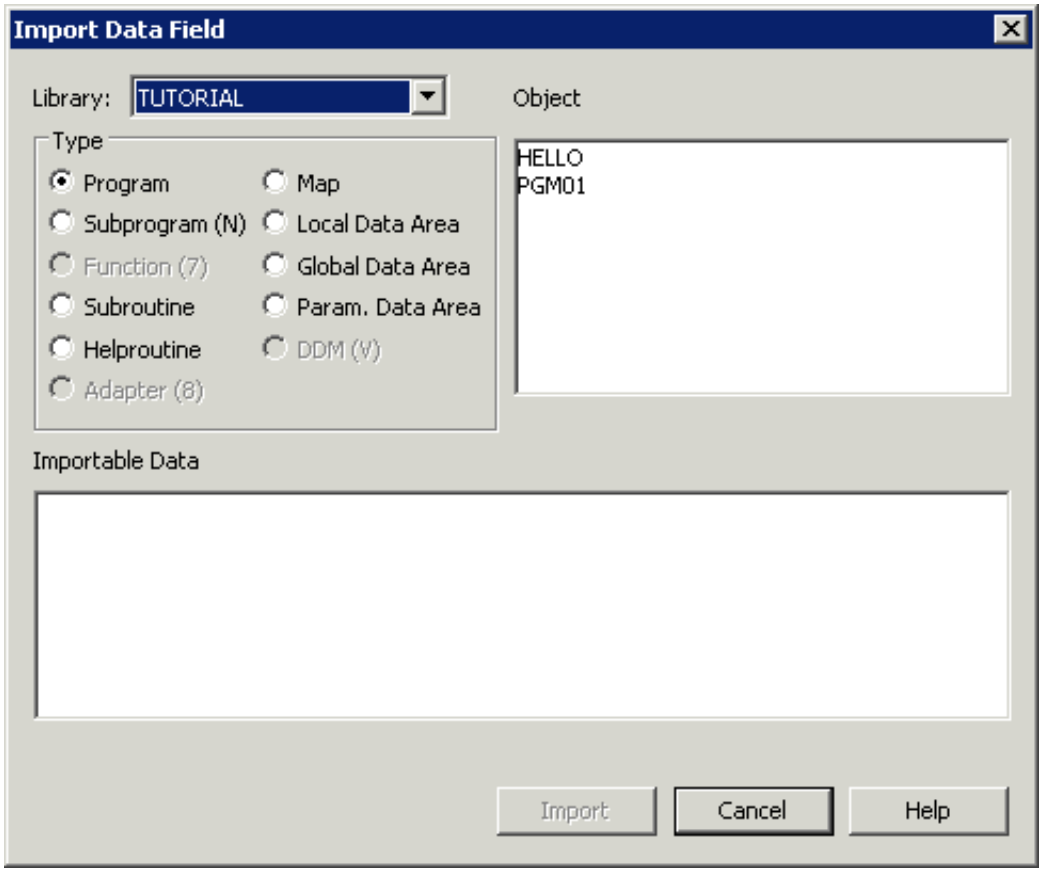
Tip:
The Windows version of Natural does not distinguish between uppercase and lowercase letters. However, when working with the mainframe version of Natural, keywords and identifiers are always entered with uppercase letters; text constants may contain lowercase letters. Therefore, if you also want to edit your programs on a mainframe, it is recommended that you always enter your program code as you would on a mainframe.

LOCAL means that the variables that you will define with the next step are local variables which apply only to this program.

To import data fields from a DDM

1. Place the cursor in the line below LOCAL.
2. From the **Program** menu, choose **Import**.

   The **Import Data Field** dialog box appears.



3. From the **Library** drop-down list box, select **SYSEXDDM**.

   When the **DDM** option button is selected, all defined DDMs are shown in the **Object** list box.

4. Select the sample DDM with the name `EMPLOYEES`.

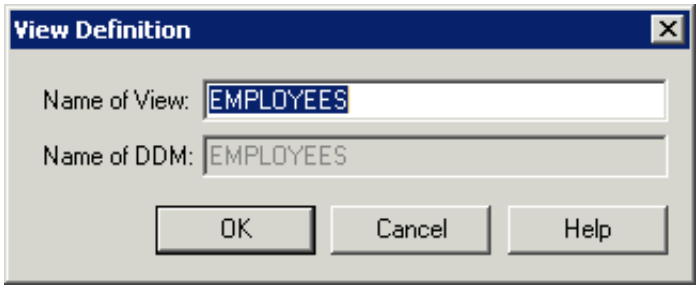
   The importable data fields are now shown at the bottom of the dialog box.

5. Press CTRL and select the following fields:

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

6. Choose the **Import** button.

   The **View Definition** dialog box appears.



By default, the name of the DDM is proposed as the view name. You can specify any other name.

7. Enter "EMPLOYEES-VIEW" as the view name.
8. Choose the **OK** button.

   The **Cancel** button in the **Import Data Field** dialog box is now labeled **Quit**.
9. Choose the **Quit** button to close the **Import Data Field** dialog box.

   The following code has been inserted in the program editor:

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

The first line contains the name of your view and the name of the database file from which the fields have been taken. This is always defined on level 1. The level is indicated at the beginning of the line. The names of the database fields from the DDM are defined at levels 2 and 3.

Levels are used in conjunction with field grouping. Fields assigned a level number of 2 or greater are considered to be a part of the immediately preceding group which has been assigned a lower level number. The definition of a group enables reference to a series of fields (this may also be only one field) by using the group name. This is a convenient and efficient method of referencing a series of consecutive fields.

Format and length of each field is indicated in parentheses. "A" stands for alphanumeric, and "N" stands for numeric.

Reading Data from a Database

Now that you have defined the required data, you will add a READ loop. This reads the data from the database file using the defined view. With each loop, one employee is read from the database file. Name, department and remaining days of vacation for this employee are displayed. Data are read until all employees have been displayed.

Note:
It may happen that an error message is displayed indicating that the transaction has been aborted. This usually happens when the non-activity time limit which is determined by Adabas has been exceeded. When such an error occurs, you should simply repeat your last action (for example, issue the RUN command once more).

To read data from a database

1. Insert the following below `END-DEFINE`:

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

   `BY NAME` indicates that the data which is read from the database is to be sorted alphabetically by name.

   The `DISPLAY` statement arranges the output in column format. A column is created for each specified field and a header is placed over the column. `3X` means that 3 spaces are to be inserted between the columns.

2. Run the program.

   The following output appears.

```
Page      1                                                  09-06-30  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
```

   As a result of the `DISPLAY` statement, the column headers (which are taken from the DDM) are underlined and one blank line is inserted between the underlining and the data. Each column has the same width as defined in the `DEFINE DATA` block (that is: as defined in the view).

   The title at the top of each page, which contains the page number, date and time, is also caused by the `DISPLAY` statement.

3. Press ENTER repeatedly to display all pages.

   You will return to the program editor when all employees have been displayed.

Tip:
If you want to return to the program editor before all employees have been displayed, press ESC.

Reading Selected Data from a Database

Since the previous output was very long, you will now restrict it. Only the data for a range of names is to be displayed, starting with "Adkinson" and ending with "Bennett". These names are defined in the demo database.

To restrict the output to a range of data

1. Before you can use new variables, you have to define them. Therefore, insert the following below LOCAL:

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

   These are user-defined variables; they are not defined in demo database. The hash (#) at the beginning of the name is used to distinguish the user-defined variables from the fields defined in the demo database; however, it is not a required character.

   INIT defines the default value for the field. The default value must be specified in pointed brackets and quotation marks.

2. Insert the following below the READ statement:

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

   Your program should now look as follows:

   ```
   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. Run the program.

The output is shown. When you press ENTER repeatedly, you will notice that you will return to the program editor after a couple of pages (that is: when the data for the last employee named Bennett has been displayed).

4. Stow the program.

You can now proceed with the next exercises: *User Input*.